National Library
of Canada

Bibliothèque nationale
du Canada

Canadian Theses Service

Services des thèses canadiennes

Ottawa, Canada
K1A 0N4

CANADIAN THESES

THÈSES CANADIENNES

NOTICE

The quality of this microfiche is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this film is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microfiche dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, examens publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de ce microfilm est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

THIS DISSERTATION
HAS BEEN MICROFILMED
EXACTLY AS RECEIVED

LA THÈSE A ÉTÉ
MICROFILMÉE TELLE QUE
NOUS L'AVONS REÇUE

NL-339 (r.86/06)

Canada

THE UNIVERSITY OF ALBERTA

FLUID FLOW MEASUREMENT USING NMR IMAGING

BY

© KNUD STEVEN KNUDSEN

A THESIS
SUBMITTED TO THE FACULTY OF GRADUATE STUDIES AND
RESEARCH IN PARTIAL FULFILMENT OF THE REQUIREMENTS FOR THE
DEGREE OF MASTER OF SCIENCE

DEPARTMENT OF APPLIED SCIENCES IN MEDICINE

EDMONTON, ALBERTA
SPRING 1987

Permission has been granted to the National Library of Canada to microfilm this thesis and to lend or sell copies of the film.

The author (copyright owner) has reserved other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without his/her written permission.

L'autorisation a été accordée à la Bibliothèque nationale du Canada de microfilmer cette thèse et de prêter ou de vendre des exemplaires du film.

L'auteur (titulaire du droit d'auteur) se réserve les autres droits de publication; ni la thèse ni de longs extraits de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation écrite.

ISBN 0-315-37767-4

THE UNIVERSITY OF ALBERTA

RELEASE FORM

NAME OF AUTHOR	KNUD STEVEN KNUDSEN
TITLE OF THESIS	FLUID FLOW MEASUREMENT USING NMR IMAGING
DEGREE FOR WHICH THESIS WAS PRESENTED	MASTER OF SCIENCE
YEAR THIS DEGREE GRANTED	SPRING 1987

Permission is hereby granted to THE UNIVERSITY OF ALBERTA LIBRARY to reproduce single copies of this thesis and to lend or sell such copies of this thesis for private, scholarly, or scientific research purposes only.

The author reserves other publication rights, and neither the thesis nor extensive extracts from it may be printed or otherwise reproduced without the author's written permission.

(SIGNED) Steven Knudsen

PERMANENT ADDRESS:

c/o Olds College
Olds, Alberta, Canada
T0M 1P0

Dated October 31 1986

THE UNIVERSITY OF ALBERTA

FACULTY OF GRADUATE STUDIES AND RESEARCH

The undersigned certify that they have read, and recommend to the Faculty of Graduate Studies and Research, for acceptance a thesis entitled FLUID FLOW MEASUREMENT USING NMR IMAGING submitted by KNUD STEVEN KNUDSEN in partial fulfilment of the requirements for the degree of MASTER OF SCIENCE.

Richard E. Snyder

Co-supervisor

Peter Allen

Co-supervisor

Z. J. Koles

J. Battista

Dated November 1, 1986

In memory of my Grandfather Rupert,

so einer Mensch werde ich nie wieder kennen.

Abstract

Within five years of the first experimental demonstration of NMR, an effect of fluid flow on the NMR signal had been observed. Flow measurement work using NMR has continued from that time through to the present. Over the past few years many investigators in the field of medical NMR imaging have pursued the measurement of blood flow.

The work in this thesis describes an approach to *in vivo* blood flow measurement using NMR imaging. The technique developed relies on the difference in the phase of the NMR signal from moving and non-moving magnetization. The change in the phase is linear with motion velocity. It can be uniquely specified by keeping the effective areas of the magnetic field gradient pulses the same on either side of a 180 degree RF pulse in a spin echo two dimensional Fourier transform imaging sequence. Constant velocity flow normal to the imaged slice was considered. All flow measurement experiments utilized gravity pressure driven tap water flowing through a simple phantom.

Difficulties with the implementation of a pulse sequence that could measure high flow rates (between 100 and 150 cm s^{-1}) led to the investigation of the effect of motion on an image. Mathematical models are presented describing the effect of constant and variable velocity flow on the two dimensional complex data set from whence images are derived. Some aspects of these models were verified by a computer simulation based on the Bloch equation. From these investigations and imaging experiments it was found that the factor limiting the flow measurement range was the slice selection gradient pulse sequence. Flow normal to the slice produces large shifts in phase when strong slice selection gradients are used. Based on the

mathematical description mentioned above and the work of van Dijk (1984) and O'Donnell (1985), a gradient pulse scheme to compensate for this effect was developed.

The resulting flow measurement imaging method is, in principle, capable of measuring flow rates well beyond those found physiologically, although only flow rates up to 15 cm s^{-1} were actually measured experimentally. Experiments have verified the linearity of the relationship between image phase and flow velocity, and the short term accuracy of the measurement technique. However, fluctuations over longer periods of time in the measured phase for a given flow rate have restricted the implementation of this technique. Recent work has indicated that changes in the effective gradient strength (i.e., how the spins react to a given gradient pulse) over time produce these variations, but this work is, as of yet, inconclusive.

Acknowledgements

I would like to thank Dr. Richard Snyder for his very good supervision through the term of the thesis project and the manuscript preparation. He has taught me much about research methods and the presentation of scientific material. On numerous occaisions his insights and questions have directed me towards solutions. Yet he has also allowed me to make my own mistakes and to learn from them. I would like to also thank my other supervisor, Dr. Peter Allen, for providing the opportunity to work in his laboratory. His thoughtful consideration of my work and my manuscript was of great benefit to me. He has helped me very much to increase my understanding of NMR.

Of the other people with whom I have been involved, a few deserve special thanks. Both Erich Treiber and Andrew Lunt have helped me immensely, Erich providing excellent tutelage on the spectrometer and Andrew with his ready help with any practical or theoretical problem that I had. A special thanks to Craig von Land for his help with the experiments and some of my software and for giving me a place to stay for these last few months. Erich, Andrew and Craig along with Chris Hanstock, Cindy Stewart and Dave Phillips have contributed very much to my and the lab's social life, making my stay very enjoyable. I am lucky to count them among my friends.

It is not really possible to properly thank all the other people who have helped me; Dan Gheorghiu, Manuel Castro, Bob Heath, Bob Morse, Dan Doran, Michael Lazar, Pete's Daughters (inspirational), Fred Brauer, Rick Hooper, Deb Kelly, Ravi Menon, Sylvia Cromwell, Dr. Koles, and Evan Peers.

The use of the Macintosh systems of Dr. Donald Boisvert and Dr. Bill Hulbert in the preparation of this manuscript is greatly appreciated.

The Alberta Heritage Foundation for Medical Reseach and the Natural Sciences and Engineering Reseach Council are gratefully acknowledged for their financial support.

Table of Contents

Chapter Page

1. Introduction 1

2. Simple Nuclear Magnetic Resonance Theory and Two Dimensional Fourier Transform Imaging 7

  2.1 Simple Nuclear Magnetic Resonance Theory 7

  2.2 The Rotating Frame of Reference 14

  2.3 Effect of Alternating Magnetic Fields 16

  2.4 Detection of the NMR Signal 21

  2.5 Relaxation Processes: T_1 and T_2 23

    2.5.1 Longitudinal Relaxation 23

    2.5.2 Transverse Relaxation 26

    2.5.3 The Two-Site-Rapid-Exchange Model of Relaxation in Biological Systems 29

  2.6 Bloch Equations 30

  2.7 Spin Echoes 32

  2.8 Two Dimensional Fourier Transform Imaging 36

  2.9 Slice Definition 46

  2.10 A Mathematical Development of the Two Dimensional Fourier Transform Imaging Technique 53

3. Flow Measurements Using NMR Imaging 60

  3.1 The Phase of the NMR Signal 62

  3.2 The Effect of Motion on a Two Dimensional Fourier Transform Image 69

  3.3 The Effect of Pulsatile Flow 76

Chapter Page

4. Computer Simulation of NMR Experiments 78
  4.1 Background 78
  4.2 Simulation Methodology 79
  4.3 Module Functional Descriptions 81
    4.3.1 Pulse Program and Bloch Equation Modules 81
    4.3.2 Signal Acquisition 86
    4.3.3 Data Manipulations 87
    4.3.4 Program Constants 87
  4.4 Simulation Constraints and Limitations 88
  4.5 Simulation Testing 89
  4.6 Flow Experiment Simulations 97
    4.6.1 Experiment I 98
    4.6.2 Experiment II 100
    4.6.3 Experiment III 102
  4.7 Conclusions and Suggestions for Improvements 104
5. Methods 106
  5.1 The Flow Phantom 106
  5.2 NMR Imaging Techniques 110
  5.3 Image Reconstruction and Data Analysis 112
6. Results and Discussion 116
  6.1 Phase Variations 116
  6.2 Dependence of Flow Measurement Range on Slice Selection Sequences 127

Chapter Page

6.3 Flow Velocity Measurements 143

6.4 A Suggested Experimental Method for Flow Velocity Measurement and Further Experiments 161

7. Conclusions 163

References 166

Bibliography 170

Appendix A 171

Appendix B 174

Appendix C 178

Appendix D 187

Appendix E 227

Appendix F 229

Appendix G 230

Appendix H 235

Appendix I 237

Appendix J 241

Appendix K 242

Appendix L 265

Appendix M 284

Appendix N 295

List of Tables

Table		Page
4.1	Some iteration and time slice effieciency comparisons.	91
4.2	Results from experiment I.	99
4.3	Results from experiment II.	101
6.1	Slice thicknesses for various gradient settings.	131
6.2	Results of the flow compensating gradient comparison test.	145
6.3	Results from test of flow measurement precision.	147
6.4	Results of flow measurement accuracy test.	149

List of Figures

Figure Page

2.1 The precession of the magnetic moment about $\underline{B}_0$. 13

2.2 The effective magnetic field in the rotating frame. 18

2.3 a) A 90° pulse in the rotating frame. 20
b) Precession in the rotating frame of the magnetization with only $\underline{B}_0$ present. 20

2.4 The changing magnetic flux through a coil due to the precession of the magnetization. 21

2.5 Laboratory and rotating frame precession of the transverse magnetization. 26

2.6 a) Dephasing of transverse magnetization. 28
b) Complete dephasing of the transverse magnetization. 28

2.7 The purposeful reduction of $\underline{B}_0$ by a linear magnetic field gradient. 33

2.8 A spin echo pulse sequence and the detected signal. 34

2.9 a) The effect of a 180 degree pulse. 35
b) Spin echo rephasing. 35
c) The completely rephased transverse magnetization. 35

2.10 The projection of spin distributions on an axis. 38

2.11 A possible 2DFT pulse sequence. 39

2.12 Two elemental magnetizations. 40

2.13 The real part of the frequency spectrum from one phase encoded scan. 43

2.14 An array of the real part of the frequency domain following the first Fourier transformation. 44

2.15 The real part of the two dimensional complex Fourier transform of the time domain data. 45

2.16 Effective magnetic field in the rotating frame. 48

Figure Page

2.17 Components of the magnetization after a rectangular 90° pulse. 49

2.18 A slice defining sequence. 53

2.19 Gradient and RF pulse timing diagram. 54

3.1 The radial dependence of a linear magnetic field gradient along the z axis. 64

3.2 An ideal magnetic field gradient pulse. 67

3.3 A typical spin echo pulse program. 70

3.4 Balanced gradient pulses. 74

4.1 A pulse program used as a model for a computer simulation. 93

4.2 The dependence of M_x on spin resonance frequency. 94

4.3 The dependence of M_y on spin resonance frequency. 94

4.4 The dependence of M_z on spin resonance frequency. 95

4.5 The dependence of M_{xy} on spin resonance frequency. 95

4.6 The dependence of the transverse magnetization phase on spin resonance frequency. 96

4.7 The pulse program modelled in experiment I. 98

4.8 The pulse program modelled in experiment II. 101

4.9 The pulse program modelled in experiment III. 103

5.1 The flow phantom. 107

5.2 A sketch of a typical image of the U-tube phantom. 114

5.3 A set of region definitions to measure the phase. 115

6.1 A phase image showing smooth variations. 117

Figure Page

6.2 a) A profile of the variation in phase along the frequency encoding direction. 119
b) A profile of the variation in phase along the phase encoding direction. 120

6.3 a) The effect of a badly mis-centred spin echo. 121
b) The appearance of a phase image when the spin echo is centred. 122

6.4 A glass tube of $CuSO_4$ doped water supported in a perspex slab. 124

6.5 a) The modulus image of the tube centred in the imaging slice. 125
b) The modulus image of the tube with the slice shifted 3 mm. 126

6.6 A pulse program to study the effect of the slice selection sequence on the phase of the NMR signal from flowing liquid. 127

6.7 Results of the z direction gradient linearity experiment. 133

6.8 The experimental setup of the 3 mm thick disk phantom. 134

6.9 a) A modulus image of the disk phantom at 57.0 cm from the front plate of the magnet. 135
b) at 57.3 cm 136
c) at 57.6 cm 137
d) at 57.9 cm 138

6.10 A curved slice that is the result of the radial dependence of the z direction magnetic field gradient. 139

6.11 A visualization of the effect of a slice selection gradient pulse along one direction and a slice refocussing gradient pulse along the opposite direction. 140

6.12 The four gradient pulse sets (used in the eddy current measurement experiment). 141

6.13 A plot of the magnetic field gradient over time between the two spheres. 142

6.14 The final flow measurement pulse program. 144

Figure		Page
6.15	The effect of different flow compensating gradient pulse amplitudes.	146
6.16	The relationship between flow velocity and the measured phase.	149
6.17	A profile across the inlet tube showing the laminar flow velocity distribution.	150
6.18	a) A phase image showing slow laminar flow.	154
	b) A phase image showing a high flow rate and turbulence.	155
6.19	a) The change in the measured phase on day zero.	157
	b) The change in the measured phase on day five.	157
6.20	The change in the measured phase over several days.	158
6.21	Results of the second experiment investigating the effect of different flow compensating gradient pulse amplitudes on the phase encoded.	160
C.1	A set of cartesian axes rotating about the laboratory frame of reference axes.	179
C.2	The rotating frame axes with respect to the laboratory frame axes.	180

Symbols and Abbreviations

$b_{i\ local}$	local magnetic field at a nuclear site
$\underline{B}$	sum of the static magnetic field and the magnetic field gradient
$\underline{B}_0$	static magnetic field
ΔB_0	magnetic field inhomogeneity
$\underline{B}(t)$	time varying magnetic field
$\underline{B}_R$, $\underline{B}_L$	counter-rotating magnetic field components
$\underline{B}_1$, $\underline{B}_1(t)$	radio frequency varying magnetic field
$\underline{B}_{eff}$	effective magnetic field
$d(t)$	windowed sampling function along x direction
$d(G_y)$	windowed sampling function along y direction
E	energy
ΔE	change in energy
EMF	electromotive force
$E_x(t)$	apodization function in x direction
$E_y(G_y)$	apodization function in y direction
f_x	frequency variable in x direction
f_y	frequency variable in y direction
F_I	imaginary part of NMR signal
F_R	real part of NMR signal
F_z	frequency of flow velocity fluctuation
FID	free induction decay
$\underline{G}_r$	linear magnetic field gradient along $\underline{r}$

G_x, G_y, G_z	magnetic field gradient along the x, y, z directions
ΔG	increment in a gradient from scan to scan
$G_{z\ comp}$, $G_{z\ compensation}$	absolute amplitude of either balanced gradient pulse lobe between the 90 and 180 degree RF pulses
$g(t)$	an arbitrary function of time
$G(f)$	the Fourier transform of g(t)
h	Planck's constant
$\underline{I}$	angular momentum operator
$\underline{i}, \underline{j}, \underline{k}$	cartesian coordinate system unit vectors
$J(\omega)$	spectral density function at angular frequency ω
J_m	mth order Bessel function of the first kind
k	i) Boltzmann's constant ii) sample index
k_x	x direction spatial frequency coordinate
k_y	y direction spatial frequency coordinate
m	i) magnetic quantum number ii) an elemental magnetization iii) number of shifted points
$\underline{\mathcal{M}}$	laboratory reference frame net magnetization
$\underline{M}$	i) laboratory reference frame net magnetization ii) rotating reference frame net magnetization
M_x, M_y, M_z	the components of $\underline{M}$
M_{xy}	the magnitude of the transverse component of $\underline{M}$
n_α	population of nuclei in lower energy state
n_β	population of nuclei in upper energy state

N	number of samples
N_x	number of samples in the x direction
N_y	i) number of samples in the y direction ii) number of scans in an imaging experiment
NMR	nuclear magnetic resonance
P	spin angular momentum magnitude
P_z	measureable component of P along the z direction
P_{free}	proportion of water in free state
P_{bound}	proportion of water in bound state
P(x,y)	transverse magnetization at (x,y) in the slice
PSF	point spread function
QPSD	quadrature phase sensitive detection
$\underline{r}$	position vector
rect	rectangular window function
RF	radiofrequency
sinc, sinc(x)	sin(x)/x function
$s(k_x,k_y)$	signal from slice in spatial frequency domain
SW	slice width (in cm)
S_{BW}	slice bandwidth (i.e., bandwidth of shaped RF magnetic field pulse in Hz)
T	i) Tesla ii) absolute temperature iii) sample period, time between samples
T_1	longitudinal relaxation time constant
T_2	transverse relaxation time constant
T_2^*	total dephasing time constant for transverse magnetization

T_R	time interval between scans
t_p	duration of a gradient pulse
t_w	duration of RF magnetic field pulse
t_x	total signal sampling time
Δt	time between samples
$\Delta t, \Delta t_z$	separation of gradient pulses in time
t_{90}, t_{180}	duration of 90 and 180 degree RF pulses
v_x, v_y, v_z	velocity components along x, y and z
V	flow velocity
V_{avg}	average flow velocity
V_{peak}	peak flow velocity
2DFT	two dimensional Fourier transform
α	lower spin energy state
β	upper spin energy state
γ	gyromagnetic ratio
$\Gamma(n)$	gamma function
$\delta(x)$	delta function
ζ	independent variable of a function
μ	magnetic moment
μ	viscosity
ν	frequency encoding direction variable

ξ	i) elemental magnetization index ii) dependent variable of a function
ρ	i) density ii) subscript to indicate rotating frame
$\rho(x,y)$	spin density at (x,y) in the slice
τ_c	correlation time for molecular motion
ϕ	phase encoding direction variable
ω_0	Larmor frequency
$\underline{\omega}$	angular velocity
$\mathbb{F}$	Fourier transform operator
f	frequency variable
$\times$	cross product operator
$\otimes$	convolution operator
$\Uparrow$	an ideal sampling function of infinite extent
$*$	multiplication operator
$\cdot$	i) between vectors - dot product operator ii) multiplication operator

1. Introduction

The discovery of Nuclear Magnetic Resonance (NMR) in 1946 by Bloch et al. (1946a, 1946b, 1946c) and Purcell et al. (1946) was soon followed by the observation of the effect of fluid flow on the NMR signal. Suryan (1951) reported that the amplitude of the signal from fluid flowing in a tube through the RF coil first increased and later decreased with increasing flow velocity. The initial increase was due to the replacement of partially saturated spins by spins strongly magnetized upstream of the coil, giving a stronger signal. As the velocity increased, the upstream spins spent less time in the magnet and were, as a result, not as strongly magnetized, therefore giving a weaker signal.

Hahn (1950) observed that motion in a liquid resulted in a decreased spin echo signal amplitude. This was further elucidated by Carr and Purcell (1954) in their study of the dependence of spin echo amplitude and phase on motion along a magnetic field gradient. They showed that the phase of successive echoes generated by a multi-echo pulse sequence was shifted by an amount proportional to the magnetic field strength and the echo number. These shifts cancelled on the even-numbered echoes. The result was a lower detected echo amplitude for the odd-numbered echoes.

These early investigations showed that the observed effect of flow on the NMR signal depended on the particular experiment performed. The sequence of radiofrequency (RF) and gradient magnetic field pulses applied to the sample uniquely determines the nature of the signal from both the static and flowing material. Investigators have used this to advantage by designing pulse sequences that differentiate the signal components of the moving magnetization from those of the spatially static magnetization.

From this work came three non-imaging flow measurement techniques. Singer, in 1959, presented work on blood flow in mouse tails. The measurements were based on changes in the NMR signal amplitude and in the longitudinal magnetization relaxation rate due to flowing blood. At this time Singer also gave the theory for " time-of-flight " measurements. A bolus of spins is irradiated by an upstream coil. At a known distance downstream the bolus is detected by a second coil and, from a measure of the time elapsed between the initial irradiation and the detection, the average velocity may be determined. Hahn (1960) proposed using pulsed magnetic field gradients to produce phase shifts in spin echoes proportional to the velocity of motion along the gradient direction. Later workers applied these techniques and combinations thereof to observe and measure the effects of motion in biological and non-biological systems (Stejskal, 1965; Arnold and Burkhart, 1965; Packer, 1969; Grover and Singer, 1971; Hayward et al., 1972; Garroway, 1974).

The development of medical NMR imaging in the early 1970s (Lauterbur, 1973; Mansfield and Grannell, 1973) added the ability to encode spatial information to the field of in vivo biological NMR research. In addition to the ability to display anatomical information, NMR imaging also provides knowledge of the NMR visible spin density and the relaxation times T_1 and T_2.

The observation of blood flow effects in images by Hinshaw et al. (1977) followed closely the inception of the imaging technique. Since that time researchers have sought to correlate the effects of blood flow on images with blood flow volumes, velocities and blood flow patterns. It was again found, as noted earlier, that the effect of flow on the NMR signal depends on the imaging pulse sequence used (Axel, 1984). For example, Singer and

Crooks (1983) used a selective saturation pulse followed by a variable delay and a conventional imaging sequence to correlate the loss in signal due to blood flow rate. Others (van Dijk, 1984; Bryant et al., 1984; O'Donnell, 1985) have used methods based on acquired phase shifts due to flow in spin echoes. The phase of the signal in the region of blood flow is compared to the phase of the signal from the surrounding stationary tissue, and the difference in phase is correlated with flow velocity. Bradley et al. (1984) have suggested an implementation of the time-of-flight method using multiple slices to trace the course and velocity of flowing blood. A similar approach used by Deimling et al. (1986) uses a refocussing pulse that is spatially independent of the slice selection pulse in a spin echo sequence to measure the distance blood has flowed since the initial excitation. Another technique (Mueller et al., 1986) quantifies flow in multi-echo images by analyzing the effects of flow on measured T_2 values. In each case, a measure of vessel area allows the calculation of flow volume.

From work based on the effect of blood flow on spin echo phase, Wedeen (1985) developed a subtraction method to provide projective images of blood vessels or NMR angiography. Two phase sensitive images are made, each at different times during the cardiac cycle. The effect of blood flow is different in each and when one is subtracted from the other, an image showing the region of blood flow results.

To date, interest in the study of the effects of motion on medical NMR images has concentrated on the study of the hemodynamics of the human cardiovascular system. If NMR flow imaging can measure velocity, volume and patterns of blood flow, it could be used to assess tissue and organ perfusion, patency of vascular grafts, and for detection of intraarterial plaque (Feinberg et al., 1985). NMR flow measurement techniques offer many advantages over existing methods. Images, and hence flow measurements, can be made non-invasively through any section of the body. There is no need to subject patients to procedures such as catheterization or radio opaque dye injection or to ionizing radiation. The risks involved with procedures such as coronary and cerebral angiography are not present in the NMR technique.

The project set forth for this thesis was the implementation of a NMR flow imaging technique on the medium bore animal unit of the University of Alberta In-Vivo NMR Facility. Techniques current in the literature were reviewed and it was thought that methods based on the generation of phase differences by flowing liquid held the most promise. Dependence of the signal from flowing material on a single NMR signal parameter and the ability to measure flow rates over a large range in a single imaging experiment were two key factors in the favour of the phase technique. In particular, the work of van Dijk (1984), Bryant et al. (1984), and O'Donnell (1985) was assessed. Their techniques rely on the influence of flow on the phase of spin echoes. The phase of the signal from the flowing blood is uniquely modified without affecting the phase of the signal from non-moving tissue.

Initial attempts to implement flow imaging were hampered by physical limitations of the spectrometer and interactions of the magnetic system during the imaging experiment. The manner in which the magnetization in the slice interacts with the pulse sequence and with the environment presented by the spectrometer is complex. For example, the dependence of the phase of the static-tissue signal on the slice defining and readout gradients for a particular pulse sequence must be considered when the sequence is designed and calibrated. Magnetic fields caused by eddy currents generated in the cryostat tubing by the switching of these gradient pulses can also affect the phase of the signal. These may be minimized, but not removed. Since the direction of flow was taken to be orthogonal to the imaging plane, the slice defining and rephasing gradient pulses affect the phase of the signal from the flowing liquid. The degree to which the phase is altered depends upon the amplitudes, durations and temporal locations of these gradient pulses. On the medium bore unit it was found that a slice defining gradient pulse followed immediately by a gradient pulse in the opposite direction (to rephase the spins) caused artifacts in the images generated by a spin echo experiment. Placing the rephasing gradient pulse after the first 180 degree radiofrequency magnetic field pulse corrected this problem. However, this placement of the rephasing gradient pulse produced large phase shifts for even slow moving spins.

To aid in the investigation of these problems and some of the corrective procedures, a computer simulation program based on the Bloch equations was developed. The results of simulations and phantom experiments using the spectrometer helped to establish a RF and gradient pulse sequence that is capable of measuring physiologically significant flow rates for flow normal to the imaging slice.

During these investigations, phantoms to simulate blood flow were constructed. These consisted of flexible and glass tubes through which tap water flowed. Due to the low viscosity of water with respect to blood, flow rates lower than the peak flow rates found physiologically had to be used in order to maintain laminar flow. As flow rates approach ranges that correspond to highly turbulent flow patterns, many velocity components are present at any given location across the flow profile (Bradley et al., 1984). Since the phase is proportional to the velocity, the phase at any given location in a region of flow in an image becomes ill-defined with turbulent flow. Accurate flow measurements were possible only when laminar flow was maintained.

The final stage of the project involved the evaluation of the current implementation of a flow imaging sequence in terms of accuracy, precision, range, and ease of operation.

2. Simple Nuclear Magnetic Resonance Theory and Two Dimensional Fourier Transform Imaging

2.1 Simple Nuclear Magnetic Resonance Theory

With any treatment of a complex experiment, such as Nuclear Magnetic Resonance (NMR) flow imaging, it is necessary to provide some background material. A complete treatment of the NMR phenomenon and NMR imaging is quite beyond the scope of this thesis. Other sources can provide much greater detail (Abragam, 1961; Slichter, 1963; Mansfield and Morris, 1982; Harris, 1983). However, a relatively simple overview of the NMR phenomenon will allow an understanding of the later work involving imaging.

Nuclei which possess angular momenta and magnetic moments will, when placed in a magnetic field, distribute themselves amongst discrete energy levels. Transitions of the nuclei between one level and the next can be induced by the application of electromagnetic radiation and give rise to the Nuclear Magnetic Resonance phenomenon. A proper description of this phenomenon requires quantum mechanics. However, since the classical equation describing the motion of a magnetic moment in a magnetic field is of the same form as the quantum mechanical equation describing the expectation value of a magnetic moment in a magnetic field, many properties of NMR may be explained classically (Slichter, 1963). Thus, this introduction to NMR phenomena will begin with a simple quantum mechanical description of the energy levels in a nuclear spin system, and then proceed with a classical treatment of magnetization.

A nucleus with unpaired nucleons possesses a net angular momentum $\underline{I}h/(2\pi)$ and, as a result, has a net magnetic moment, $\underline{\mu}$, collinear with it. $\underline{I}$ is the angular momentum operator and I the spin quantum number or spin of the nulceus. I may be an integer or a half-integer. Planck's constant is *h*. The spin angular momenta and the orbital angular momenta of the individual nucleons combine vectorially to give the observed resultant angular momentum for the nuclear species. The contribution of the orbital angular momenta is determined by the distribution of the nucleons in their orbital energy levels. In the nuclear ground state, the filling of these levels depends on the number of nucleons present in the nucleus, or, in other words, the particular nuclear species. The total angular momentum of any isolated particle can not have any arbitrary magnitude. It can assume only discrete values and is therefore said to be quantized. The magnitude of the spin angular momentum is given by

$$P = \frac{h}{2\pi}\sqrt{I(I+1)}$$

2.1)

The measureable components of $\underline{I}h/(2\pi)$ in a direction defined by an interacting magnetic field, say along z, are given by the magnetic quantum number, m, in the equation

$$P_z = \frac{h}{2\pi}m$$

2.2)

Quantum mechanics shows that the allowed values of m are $m = I,\ I-1, I-2, \ldots, -I$.

In a similar fashion, the observed magnetic moment $\underline{\mu}$ is dependent on the particular nuclear species. This is because the relationship between the magnetic moment vectors and the angular momentum vectors is not the same for the two cases of spin and orbit. Thus, two nuclear species with the same $\underline{I}h/(2\pi)$ will not have the same magnetic moment. The measureable component of a magnetic moment is defined to be that component parallel to a measureable component of the angular momentum vector. Thus, the observable nuclear magnetic moment, $\underline{\mu}$, is given by

$$\underline{\mu} = \gamma \frac{h}{2\pi} \underline{I} \tag{2.3}$$

where γ is the gyromagnetic ratio (radians seconds^{-1} Tesla^{-1}) and, following the argument given in the previous paragraphs, is unique for each nuclear species. Such nuclei, in a static magnetic field $\underline{B}_0$, are allowed energies corresponding to

$$E = -\gamma \frac{h}{2\pi} B_0 m \qquad m = I,\ I-1, \ldots, -I \tag{2.4}$$

For a spin $I = 1/2$, there are two discrete energy states. Therefore, the energy difference between these two states is

$$\Delta E = \gamma \frac{h}{2\pi} B_0\, 1/2 - \left(-\gamma \frac{h}{2\pi} B_0\, 1/2 \right) = \gamma \frac{h}{2\pi} B_0 \tag{2.5}$$

A simple model suggests that transitions of the spins between these states can be brought about through the application of electromagnetic radiation of angular frequency ω_0 that satisfies the Bohr frequency condition of

$$\Delta E = \frac{h}{2\pi}\omega_0 \qquad 2.6)$$

From equations 2.5) and 2.6) it is seen that

$$\omega_0 = \gamma B_0 \qquad 2.7)$$

Equation 2.7) is a statement of the resonance condition. ω_0, the resonance frequency, is called the Larmor frequency. Electromagnetic energy applied at this frequency will cause a transition of spins between the lower and upper energy states. For the rest of this thesis, only nuclei with spin $I = 1/2$ will be considered. In addition, following convention, the lower energy state is defined as α and the upper energy state as β.

In a static magnetic field the relative spin populations in these two states are governed by the amount of thermal energy available in the external environment. Since energy exchange between the nuclear spin system and the external environment is a thermally balanced process, the spin populations follow a Boltzmann distribution. Letting n_α be the population of nuclei in the lower energy state and n_β the population in the upper state, the ratio of the final equilibrium populations is given by

$$\frac{n_\alpha}{n_\beta} = e^{\frac{\Delta E}{kT}} = e^{\gamma \frac{h}{2\pi} \frac{B_0}{kT}}$$

2.8)

where k is Boltzmann's constant and T is the absolute temperature. At room temperatures this ratio is very nearly one. For example, considering hydrogen nuclei, which have a spin I = 1/2 and a gyromagnetic ratio of $\gamma = 2\pi \cdot 100 \cdot 10^6/2.35$ (radians seconds^{-1} Tesla^{-1}), in a magnetic field of $B_0 = 2.35$ Tesla and at a temperature of T = 300K, equation 2.8) yields $n_\alpha/n_\beta = 1.000016$. If there were 2 000 000 nuclei in the upper energy level, β, then there would be 2 000 032 nuclei in the lower energy level, α. This difference yields a very small, but detectable, net magnetization parallel to $\underline{B}_0$.

While a simple model for spin transitions between energy levels has been useful in introducing simple resonance, it is not a complete description of the NMR phenomenon. The behaviour of the nuclear magnetization at frequencies other than the Larmor frequency indicates that discrete transitions between stationary energy states is not an appropriate model for NMR. It is still necessary that the applied electromagnetic radiation meet the resonance condition exactly for a complete inversion of the nuclear spin populations. However, at frequencies other than the Larmor frequency the probability of finding a spin in a state different from that in which it started is always less than one, but not always zero. The probability of finding a spin in the higher energy state, β, is a smoothly varying $\sin^2$ function of the time it is irradiated and also dependent on a shape function relating the frequency of the applied electromagnetic radiation with the Larmor frequency (Abragam, 1961).

The energy level treatment of the NMR phenomenon yields information only about components of $\underline{\mu}$ in the direction of $\underline{B}_0$. The fact that there are finite expectation values for components of $\underline{\mu}$ orthogonal to $\underline{B}_0$ (Slichter, 1963) suggests that a better model would consider the NMR system in terms of the magnetization. Therefore, from this point on, the classical form of the equations of motion of a magnetic moment will be utilized. As stated before, this approach is adequate in the context of this thesis. A reconciliation of the classical and quantum mechanical forms of the equations of motion may be found in Slichter (1963).

Classically, the motion of a magnetic moment, $\underline{\mu}$, is given by the equation for the torque exerted on the moment by a magnetic field $\underline{B}_0$. The torque exerted on the moment is equated with the rate of change of angular momentum giving

$$d\left(\frac{h}{2\pi}\underline{I}\right)/dt = \underline{\mu} \times \underline{B}_0 = \mu B_0 \sin\theta$$

Since $\underline{\mu} = \gamma h/(2\pi)\, \underline{I}$ (equation 2.3),

$$d\underline{\mu}/dt = \underline{\mu} \times (\gamma \underline{B}_0) \qquad 2.9)$$

The changes in $\underline{\mu}$ in time are defined by the cross product of $\underline{\mu}$ and the magnetic field $\underline{B}_0$ and are, therefore, perpendicular to both $\underline{\mu}$ and $\underline{B}_0$. This results in a precession of $\underline{\mu}$ about $\underline{B}_0$ in a cone of half angle θ. This is the angle between the moment and $\underline{B}_0$ (Figure 2.1).

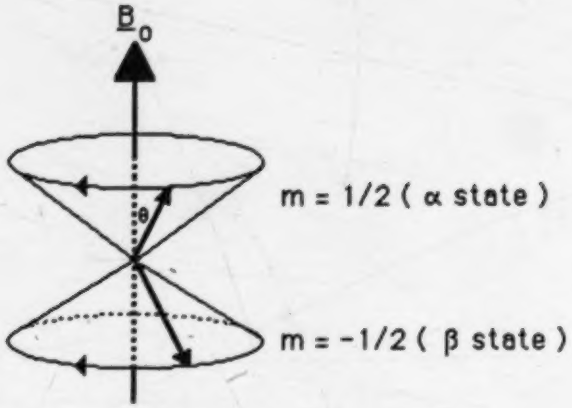


Figure 2.1 The precession of the magnetic moment about $\underline{B}_0$ according to equation 2.9).

The direction of precession is anti-clockwise when viewed along $\underline{B}_0$ and is taken to be positive for a positive gyromagnetic ratio (γ).

At this time it is useful to change from considering the changes in time for one spin to the changes in time for an ensemble of spins. The vector quantity $\underline{M}$ is referred to as the net magnetization and is given by

$$\underline{M} = \Sigma_i \, \underline{\mu}_i \qquad 2.10)$$

which is the vector sum of a large number of identical nuclear magnetic moments over a unit volume. In thermal equilibrium there is an excess of nuclei in the lower energy state or α state. Since nuclear magnetic moments in this state are oriented parallel to $\underline{B}_0$, the net magnetization, $\underline{M}$, at thermal equilibrium will be parallel to $\underline{B}_0$. If the interactions among the spins of this ensemble can be neglected, equation 2.9) can be written

$$d\underline{M} / dt = \underline{M} \times (\gamma \underline{B}_0) \qquad 2.11)$$

This magnetization vector will undergo changes in orientation and magnitude due to various magnetic field perturbations. The torque exerted on the magnetization causes it to process at a rate

$$\omega_0 = \gamma B_0 \qquad 2.12)$$

which is the same frequency as that needed to induce a change in the spin populations between the two energy states. The direction of this precession is the same as that for the individual spins that compose $\underline{M}$.

2.2 The Rotating Frame of Reference

At this point it is useful to change frames of reference from the fixed, or laboratory frame, in which the magnetization processes about $\underline{B}_0$ at a rate ω_0, to a frame rotating in the same direction as the magnetization and near or at ω_0. This simplifies the analysis of the motion of the magnetization in the presence of magnetic fields additional to $\underline{B}_0$. Since this frame rotates about the direction of $\underline{B}_0$, the static field vector still appears constant in time. However, if the frame rotates at exactly ω_0, then the magnetization components orthogonal to $\underline{B}_0$ also appear to be fixed with respect to the rotating axes.

The mathematical argument proceeds as follows (Farrar and Becker, 1971). Consider the magnetization as a vector function of time, $\underline{M}(t)$, separated into components along a set of right-handed cartesian coordinates. That is,

$$\underline{M}(t) = \underline{i}\, M_x(t) + \underline{j}\, M_y(t) + \underline{k}\, M_z(t) \qquad 2.13)$$

where $\underline{i}$, $\underline{j}$ and $\underline{k}$ are orthogonal unit vectors which can at most rotate with an instantaneous angular velocity of $\underline{\omega}$. This rotation is written

$$\partial \underline{i} / \partial t = \underline{\omega} \times \underline{i},\ \partial \underline{j} / \partial t = \underline{\omega} \times \underline{j},\ \partial \underline{k} / \partial t = \underline{\omega} \times \underline{k}, \qquad 2.14)$$

The time derivative of $\underline{M}(t)$ is then

$$\begin{aligned} d\,\underline{M} / dt &= \underline{i}\,\partial M_x/\partial t + M_x\,\partial \underline{i}/\partial t + \underline{j}\,\partial M_y/\partial t + M_y\,\partial \underline{j}/\partial t \\ &\quad + \underline{k}\,\partial M_z/\partial t + M_z\,\partial \underline{k}/\partial t \\ &= \underline{i}\,\partial M_x/\partial t + \underline{j}\,\partial M_y/\partial t + \underline{k}\,\partial M_z/\partial t \\ &\quad + \underline{\omega} \times (\,\underline{i}\,M_x + \underline{j}\,M_y + \underline{k}\,M_z\,) \\ &= \partial \underline{M}/\partial t + \underline{\omega} \times \underline{M} \qquad 2.15) \end{aligned}$$

The term $\partial\underline{M}/\partial t$ represents the time rate of change of $\underline{M}$ with respect to the coordinate system $\underline{i}$, $\underline{j}$ and $\underline{k}$. Combining equations 2.11) and 2.15) yields

$$\begin{aligned} \partial \underline{M}/\partial t + \underline{\omega} \times \underline{M} &= \underline{M} \times (\,\gamma\,\underline{B}_0\,) \\ \partial \underline{M}/\partial t &= \underline{M} \times \gamma(\,\underline{B}_0 + \underline{\omega}/\gamma\,) \qquad 2.16) \end{aligned}$$

The term $\underline{B}_0 + \underline{\omega}/\gamma$ can be thought of as the effective magnetic field, $\underline{B}_{eff}$. Equation 2.16) states that in the rotating frame of reference, $\underline{M}$ will precess about $\underline{B}_{eff}$ at a rate given by γB_{eff}. This is analogous to the precession of $\underline{M}$ about $\underline{B}_0$ in the laboratory frame of reference. $\underline{\omega}$ can be defined so that the reference frame rotates near or at the Larmor frequency. If this new frame of reference is to rotate in the same sense and at the same rate, ω_0, as the precession of the magnetization, $\underline{\omega}$ is set to $-\gamma\,B_0\,\underline{k}$. This gives $\underline{B}_{eff} = 0$ and $\partial\underline{M}/\partial t = 0$. In other words, $\underline{M}$ remains fixed with respect to $\underline{i}$, $\underline{j}$ and $\underline{k}$, and these axes rotate at $\underline{\omega} = -\gamma B_0 \underline{k}$ with respect to the laboratory frame of reference.

2.3 Effect of Alternating Magnetic Fields

Up to this point, only a static magnetic field has been considered. As mentioned before, the application of electromagnetic radiation at or near the Larmor frequency will cause a change in the spin energy level populations. The component of this radiation that induces such a change is the time varying magnetic field. This field, given by $\underline{B}(t) = \underline{i}2B_1(t)\cos\omega t$, may be considered to comprise two rotating components, each of amplitude B_1, one rotating clockwise, the other anti-clockwise. That is,

$$\underline{B}(t) = \underline{i}2B_1(t)\cos\omega t = \underline{B}_R + \underline{B}_L$$

$$\underline{B}_R = B_1\,(\,\underline{i}\cos\omega t + \underline{j}\sin\omega t\,) \tag*{2.17)}$$

$$\underline{B}_L = B_1\,(\,\underline{i}\cos\omega t - \underline{j}\sin\omega t\,) \tag*{2.18)}$$

One component will rotate in the same sense as the precession of the magnetization and the other in the opposite sense. Near resonance, the counter-rotating component can be ignored. Furthermore, since $\underline{B}_R$ can be obtained by changing the direction of rotation of $\underline{B}_L$ from ω to $-\omega$, there is no loss in generality if only $\underline{B}_R$ is considered. With this in mind, the rotating magnetic field can be written as

$$\underline{B}_1(t) = B_1\,(\,\underline{i}\cos\omega t + \underline{j}\sin\omega t\,) \tag*{2.19)}$$

Now, including both $\underline{B}_1(t)$ and $\underline{B}_0 = B_0\underline{k}$, equation 2.11) becomes

$$d\underline{M}/dt = \underline{M} \times \gamma [\underline{B}_0 + \underline{B}_1(t)] \qquad 2.20)$$

which is the equation of motion of magnetization in the laboratory frame of reference. Transforming to the rotating frame of reference, and arbitrarily taking the x axis of this frame to be along $\underline{B}_1$, equation 2.20) becomes

$$\partial \underline{M}/\partial t = \underline{M} \times [\underline{k} (\omega + \gamma B_0) + \underline{i}\, \gamma B_1]$$

$$= \underline{M} \times \gamma \underline{B}_{eff} \qquad 2.21)$$

where

$$\underline{B}_{eff} = \underline{k}(B_0 + \omega/\gamma) + \underline{i} B_1 \qquad 2.22)$$

From equation 2.21) it is seen that in the rotating frame the magnetization experiences an effective magnetic field, $\underline{B}_{eff}$. The magnetization will precess in a cone of half angle θ about the direction of $\underline{B}_{eff}$ at an angular frequency γB_{eff} (Figure 2.2), where θ is given by the following equation

$$\theta = \tan^{-1} [B_1/(B_0 + \omega/\gamma)] \qquad 2.23)$$

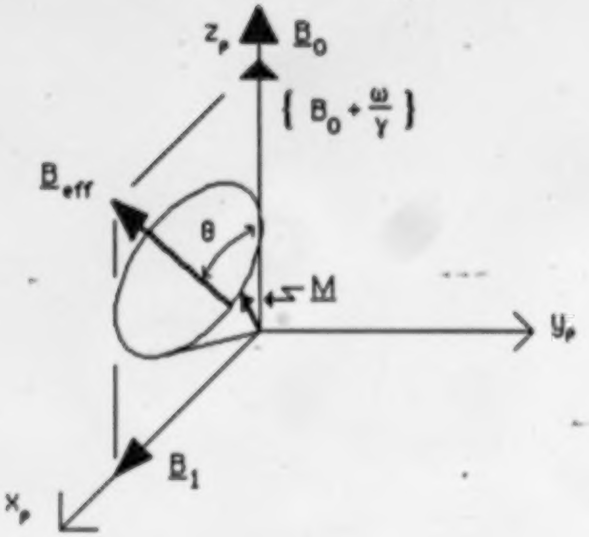


Figure 2.2 The effective magnetic field and the precession of the magnetization about it in the rotating frame. The subscript ρ refers to the rotating frame of reference.

Note that there is no net absorption of energy from $\underline{B}_1$, but rather an alternate receiving and returning of energy as the magnetization tilts away from and then back to $\underline{B}_0$. If the resonance condition is met exactly (i.e., the frequency of the electromagnetic radiation is $\omega_0 = \gamma B_0$), then $\underline{B}_{eff} = \underline{i}\, B_1$. Magnetization that is initially parallel to $\underline{B}_0$ will then precess in the y-z plane of the rotating frame. That is, the torque exerted on the magnetization by B_1 along the x axis will cause it to precess in a plane perpendicular to the direction of $\underline{B}_1$. The angle through which the magnetization will precess in the y-z plane is governed by the duration of the $\underline{B}_1$ field. Designating this time as t_w, the magnetization is seen to rotate through an angle of $\theta = \gamma B_1 t_w$. A pulse of duration t_w such that $\theta = \pi$ would simply invert the magnetization. This is known as a " π " or " 180 degree " pulse. Likewise, a pulse of duration t_w such that $\theta = \pi/2$ is known as a " $\pi/2$ " or " 90 degree " pulse.

The radio frequency electromagnetic radiation is known as " RF ". So, following a 90 degree RF pulse applied along the positive x axis (i.e., $\underline{B}_1$ is along the x axis of the rotating frame), the magnetization, $\underline{M}$, would remain along the y axis of the rotating frame. Hence, it would precess in the laboratory frame, normal to the static field $\underline{B}_0$ (Figure 2.3).

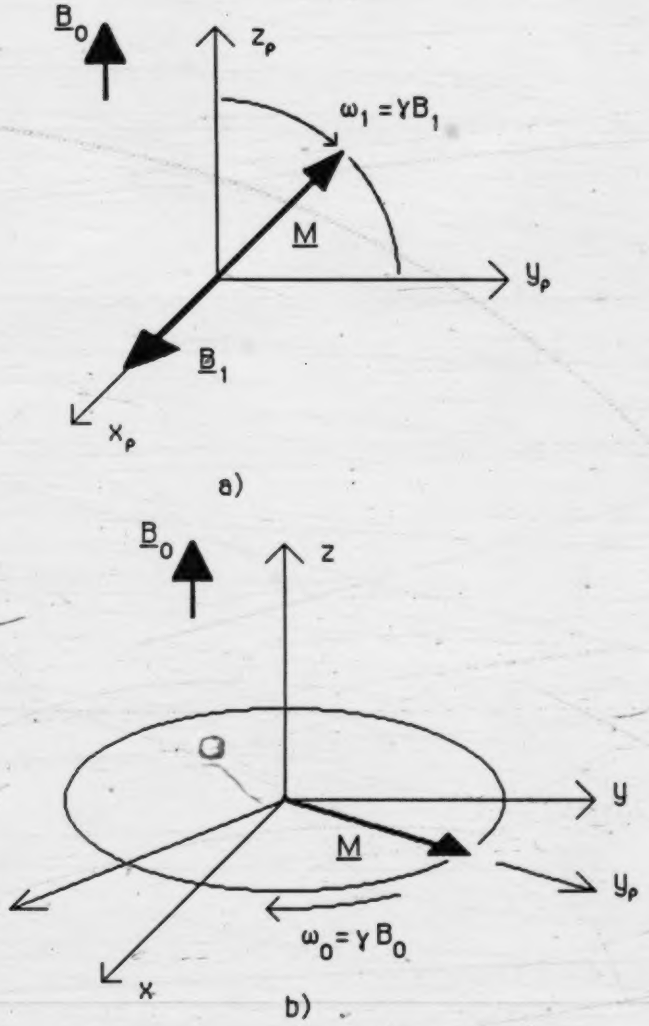

$\underline{B}_0$
z_ρ
$\omega_1 = \gamma B_1$
$\underline{M}$
y_ρ
$\underline{B}_1$
x_ρ
a)
$\underline{B}_0$
z
y
$\underline{M}$
y_ρ
x_ρ
$\omega_0 = \gamma B_0$
x
b)

Figure 2.3 a) An on-resonance ($\omega = \omega_0$) 90° pulse along the x axis in the rotating frame and its effect on the magnetization. b) Precession of the magnetization in the laboratory frame with only the static field, $\underline{B}_0$, present. In a frame rotating at ω_0, $\underline{M}$ appears static.

2.4 Detection of the NMR Signal

Faraday's law of induction states that the induced electromotive force, EMF, in a circuit is proportional to the rate at which the magnetic flux through the circuit is changing. As mentioned earlier, any net component of the magnetization in a plane orthogonal to $\underline{B}_0$ will precess in the laboratory frame of reference about the direction of $\underline{B}_0$. If a coil of conducting wire is placed so that its axis lies in this plane, there will be a change of the flux through this coil with time as the magnetization precesses (Figure 2.4). Hence an EMF will be induced in the coil. If the coil is a closed circuit, a measureable current will flow. This is the origin of the measured NMR signal.

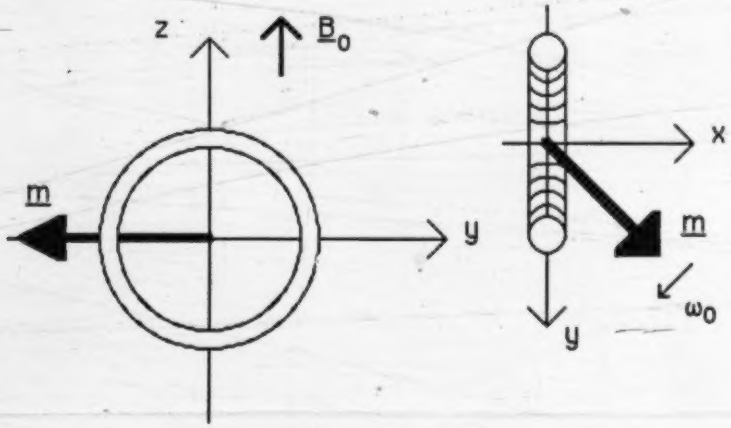


Figure 2.4 The changing magnetic flux through a coil due to the precession of the magnetization.

Since the changing magnetic flux is due to a precessing magnetization, the induced EMF, and hence the NMR signal, will be sinusoidal in nature. If the spins comprising the magnetization did not interact with anything and were in a perfectly homogeneous static magnetic field, the NMR signal would last indefinitely. In reality, however, the interactions of the spins with themselves and with their environment causes a decay of the signal. Inhomogeneities in the magnetic field cause differing local magnetic fields at the sites of the elemental magnetizations. Hence, there can be different precessional rates for these magnetizations and, as a result, sinusoidal signals of different frequencies detected. The observed decaying sinusoids are termed a " free induction decay ", or FID. The acquired time domain signal is usually analyzed in the frequency domain. A Fourier transformation of the FID yields the frequency domain spectrum of the signal.

For example, if the sinusoidal signal of frequency f from a single magnetization in a homogeneous magnetic field did not decay, the Fourier transform of it would yield a delta function at $+f$ and $-f$. The multiplication of this sinusoidal signal with a decay envelope, say a decaying exponential, yields, after the Fourier transformation, a delta function at $+f$ and $-f$ convolved with the Fourier transform of a decaying exponential, which is a Lorentzian envelope. The width of the Lorentzian envelope depends on the rate of the exponential decay. Since the convolution of a delta function with any other waveform yields only the waveform, it is seen that the Fourier transform of a single frequency, exponentially decaying FID is a pair of Lorentzian envelopes at $+f$ and $-f$.

Normally, the NMR signal is detected in quadrature. That is, the components of the magnetization which are in-phase and out-of-phase with the reference electromagnetic radiation are both simultaneously recorded. The components of the signal at frequencies above and below ω_0 are spread in a finite band about ω_0 in the frequency domain. Normally this band is shifted to be centred about $2\omega_0$ and 0 radians per second by multiplying the signal from the detection coil by a reference frequency at ω_0. This signal is then low pass filtered to retain only the components about 0 radians per second. This scheme results is a complex data point for each sample and, after a complex Fourier transformation, real and imaginary components in a frequency spectrum about $\omega = 0$. This frequency spectrum can be considered in terms of a real and an imaginary spectrum or in terms of a magnitude ($[\text{real}^2 + \text{imaginary}^2]^{1/2}$) spectrum and a phase ($\theta = \arctan[\text{imaginary/real}]$) spectrum. This technique is known as Quadrature Phase Sensitive Detection (QPSD).

2.5 Relaxation Processes: T_1 and T_2

2.5.1 Longitudinal Relaxation

The re-establishment of the Boltzmann population distribution amongst the stationary spin energy states in a static magnetic field, $\underline{B}_0 = B_0\underline{k}$, following a disruption of this population is known as longitudinal relaxation. It is called longitudinal relaxation because the recovery of the magnetization is parallel to $\underline{B}_0$. For example, a change in the thermal equilibrium spin energy state populations will result when electromagnetic radiation at the Larmor frequency is applied. There will be a net absorption of energy as nuclei change from the lower energy state to the upper state. Once the electromagnetic radiation is removed, the re-establishment of thermal equilibrium requires a net number of transitions of nuclei from the upper to the lower energy state. These nuclei give up energy in this process. Hence, there must be some other system to which the nuclei are coupled which will accept this energy. This system is the thermal environment of the nuclei and is referred to as the " lattice " (regardless of whether it is in a solid, liquid, or gaseous state). The exchange of energy between the lattice and the spin system has led to the term " spin-lattice relaxation ".

The change in time of a spin energy state population (n_α or n_β) can be written as a first order rate equation that includes terms giving the probability of a spin transition from one energy state to another. It can be shown (Slichter, 1963) that this equation leads to the differential equation

$$dM_z(t)/dt = k[\, M_0 - M_z(t) \,] \qquad 2.24)$$

which describes the re-establishment of the magnetization parallel to $\underline{B}_0$. The solution of equation 2.24) is

$$M_z(t) = M_0 + Ae^{-kt} \qquad 2.25)$$

where A is a constant of integration. The term k is conventionally defined in terms of a characteristic relaxation time T_1 (i.e., $T_1 = 1/k$). For example, if an initially unmagnetized sample is placed in a magnetic field, the magnetization process is described by an exponential rise to the thermal equilibrium magnetization along z, M_0. In this case the above equation would be

$$M_z(t) = M_0(\, 1 - e^{-t/T1} \,) \qquad 2.26)$$

As stated earlier, transitions of the spins between spin energy levels can only be induced by time varying magnetic fields. All mechanisms for longitudinal relaxation involve the modulation in time of the local magnetic field at a nuclear site. For example, this modulation can be due to the thermal motion of other nuclei. As with the application of radio frequency electromagnetic radiation, the frequency at which these transitions occur is the Larmor frequency, ω_0.

The rate at which the local magnetic fields are modulated is related to the spectral density function, $J(\omega)$, of the thermally generated motion of the nuclei. For a stationary random process, $J(\omega)$ can be expressed as a Debye spectral density,

$$J(\omega) = \frac{\tau_c}{1 + \omega^2\tau_c^2} \qquad 2.27)$$

where τ_c is known as the correlation time. τ_c can be thought of as the " average " time between changes in the direction of motion for a molecule. The rate of change in molecular motion depends on the value of τ_c; if τ_c is short, the change in motion is rapid, if it is long, the change is slow. A nucleus possessing a magnetic moment in a moving molecule will act to modulate the local magnetic field in time. Hence τ_c is related to the frequency of the local magnetic field modulation. It can be shown (Abragam, 1961) that longitudinal relaxation is most efficient when the product $\omega_0\tau_c$ is approximately equal to 0.62. This statement implies that the relaxation becomes more efficient as the frequency of motion " tunes " in with the resonant frequency.

2.5.2 Transverse Relaxation

The process which describes the time variation of the components of the magnetization in a plane normal to the direction of $\underline{B}_0$, the so called transverse plane, is referred to as transverse relaxation. Some authors refer to this as " spin-spin relaxation ".

If the field $\underline{B}_0$ were perfectly static and homogeneous and the spins within the sample did not interact, then all the components of the magnetization would experience only $\underline{B}_0$. The transverse components of the magnetization would precess at the Larmor frequency, $\omega_0 = \gamma B_0$, indefinitely (Figure 2.5). This would mean that the magnitude of M_{xy}, the transverse magnetization, would stay constant in time and, hence, so would the signal induced in the detection coil.

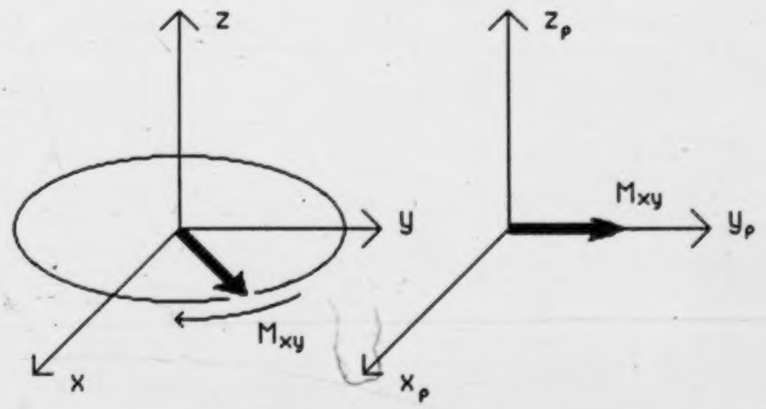


Figure 2.5 The laboratory frame precession of the transverse plane component of the magnetization for the case where the spins in the magnetization all experience the same magnetic field. In the rotating frame (subscript ρ) the magnetization appears static.

This is, however, not the case. Each nucleus in the sample experiences the main field $\underline{B}_0$ plus a local magnetic field contribution due to other nuclear magnetic moments in the immediate vicinity and to electronic hyperfine interactions. These local fields, designated here by a small b, vary with the local structural environment, the temperature and the rate of motion of the neighbouring spins. As a result, the precession frequencies of the spins comprising the magnetization will be

$$\omega_i = \omega_0 \pm \gamma b_{i\ local} \qquad 2.28)$$

where i refers to the ith spin in the magnetization vector. A distribution of local magnetic fields within the magnetization volume will lead to a distribution of precessional frequencies. That is, some spins will precess at ω_0, some will precess at frequencies less than ω_0, and some will precess at frequencies higher than ω_0. When viewed in a frame rotating at ω_0, the components of the magnetization will be seen to " fan out " or dephase with time. In this frame, components in a field less than B_0 will precess negatively while those in a field greater than B_0 will precess in a positive direction. In time, the magnetization will completely dephase, i.e., there will be no net magnetization vector in the transverse plane. This process differs from longitudinal relaxation in that there is no energy exchange between nuclei and the lattice. In liquid systems transverse relaxation can be modelled with the first order rate equation

$$dM_{xy}(t)/dt = -k\, M_{xy} \qquad 2.29)$$

The solution of this equation gives an exponential decay term, e^{-kt}. Again, k is taken to be related to a characteristic time, T_2 ($T_2 = 1/k$). This decay process modulates the sinusoidal NMR signal to give the characteristic decay as seen in the free induction decay.

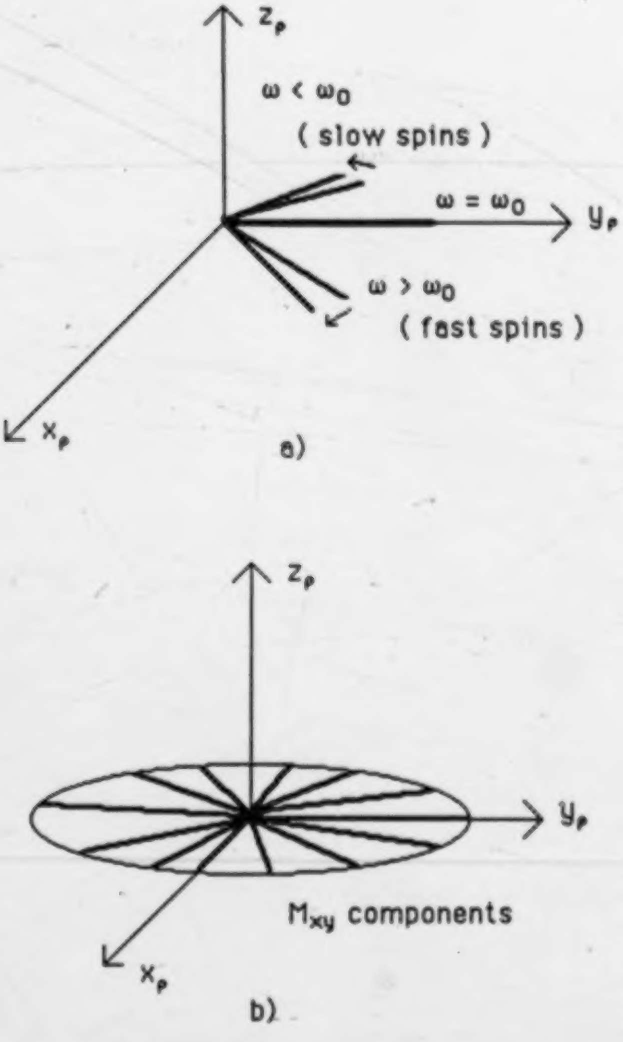

z_ρ
$\omega < \omega_0$
(slow spins)
$\omega = \omega_0$
y_ρ
$\omega > \omega_0$
(fast spins)
x_ρ
a)
z_ρ
y_ρ
M_{xy} components
x_ρ
b)

Figure 2.6 a) Components of the magnetization in the transverse plane beginning to dephase in the rotating frame.
b) Eventually (after ~ 4 T_2) the magnetization is completely dephased.

In rigid environments (e.g. crystals) the local magnetic fields produced by the interaction of the spins are very large ($\approx 10^{-4}$ Tesla) and hence, transverse relaxation is very rapid (T_2 is short, on the order of tens of microseconds). In biological samples the spins are much more mobile and the local fields time average to smaller values (10^{-7} Tesla or less). This gives a much longer transverse relaxation time (on the order of tens of milliseconds). As with longitudinal relaxation, the efficiency of the relaxation is related to the rate of motion.

2.5.3 The Two-Site-Rapid-Exchange Model of Relaxation in Biological Systems

The two magnetization relaxation processes described modify the NMR signal and are, therefore, important parameters giving information about the sample (Mansfield and Morris, 1982).

An important model that describes the effect of relaxation for protons (spin $I = 1/2$) in water molecules in tissue is the two-site-rapid-exchange model (Zimmerman and Brittin, 1957). It is based on the assumption that water in tissue exists in two environments or phases. Water may be free (unassociated with other molecules), in which case it has the NMR properties (namely T_1 and T_2) similar to a water solution of cellular salts; or it may be " bound " to or " associated " with a macromolecule (for instance, through hydrogen bonds).

As mentioned before, the interaction of the spins with local magnetic fields leading to longitudinal and transverse relaxation, depends upon molecular motion at the Larmor frequency, ω_0. Free protons move very rapidly and have a small motional component at ω_0. Hence, their relaxation processes are inefficient. The macromolecules move much more slowly and have a larger motional component at ω_0. The relaxation of protons associated with these molecules is therefore more efficient. The other factor affecting the observed relaxation rate is the exchange of water molecules between the two phases. The water molecules exchange rapidly between the unassociated or free state and the associated or macromolecule bound state. The observed relaxation rate is a weighted average of the relaxation rates for both states and can be modelled by

$$1/T_{obs} = P_{free} \cdot 1/T_{free} + P_{bound} \cdot 1/T_{bound} \qquad 2.30)$$

where T refers to the time constant of the particular relaxation process (longitudinal or transverse). P_{free} and P_{bound} refer to that proportion of water in the free and bound states repectively.

2.6 Bloch Equations

To this point the classical equations have been developed to include the motion of the magnetization in a static magnetic field, the effect of alternating magnetic fields and the relaxation processes characterized by the time constants T_1 and T_2. In 1946 Felix Bloch (1946c) proposed a set of simple equations based on phenomenological arguments that have for the most part in the case of liquid samples, correctly described the properties of the magnetization in magnetic fields. The equations were justified heuristically as follows.

First, it has been shown that the equation of motion of magnetization in a static, homogeneous magnetic field is given by equation 2.11)

$$d\underline{M}/dt = \underline{M} \times \gamma\underline{B}_0$$

Second, the recovery of the magnetization to its thermal equilibrium value, $M_z = M_0$, in a static magnetic field $\underline{B}_0$ can often be described with good accuracy by the equation

$$dM_z/dt = (M_0 - M_z)/T_1 \qquad 2.31)$$

where T_1 is referred to as the longitudinal relaxation time.

Third, the decay of the signal induced by transverse components of the magnetization (components normal to the static field, $\underline{B}_0$) can often be expressed by the equations

$$dM_x/dt = -M_x/T_2 , \quad dM_y/dt = -M_y/T_2 \qquad 2.32)$$

Note that this is just a decomposition of equation 2.29).

Bloch assumed that in the presence of an applied magnetic field (the sum of the static and time varying fields) the motion due to relaxation could be superposed on the motion of the free spins. This led to the equation

$$d\underline{M}/dt = \underline{M} \times \gamma\underline{B} - (M_x\underline{i} + M_y\underline{j})/T_2 + (M_0 - M_z)\underline{k}/T_1$$

2.33)

known as the Bloch Equation, where $\underline{i}$, $\underline{j}$ and $\underline{k}$ are the unit vectors of the laboratory frame of reference. This equation decomposes into a set of three equations for each of the x, y and z components of the magnetization $\underline{M}$. Limited to the work described in this thesis (as only liquid systems are considered), these equations accurately describe the interaction of the macroscopic magnetization with static and time varying magnetic fields.

2.7 Spin Echoes

In section 2.5, modulation of the local magnetic field experienced by a nucleus was said to be due to the thermally generated motion of other nuclear magnetic moments. This modulation of local magnetic fields causes dephasing of the transverse magnetization components or, transverse relaxation. Since thermally generated motion is a random process, the fluctuations of the local magnetic fields in time are also random. A random process cannot be reversed, hence transverse relaxation is an irreversible process.

However, dephasing can also occur because of other field inhomogeneities. One such cause of field inhomogeneity is the application of magnetic field gradients. These gradients vary the strength of $\underline{B}_0$ in space by supplying an additional magnetic field component (Figure 2.7). As a result of this generated inhomogeneity, the precession frequency varies in the direction of the gradient. Inhomogeneities can also be caused by imperfect magnet and coil construction, eddy currents generated in the cryostat by pulsed magnetic fields, etc. These inhomogeneities add to the local magnetic fields experienced by the spins. This increases the rate of dephasing of the transverse magnetization and hence, the observed rate of the transverse relaxation. This increased rate of decay is characterized by a time constant T_2^*, which includes contributions from the true transverse relaxation process and from magnetic field inhomogeneity. Thus,

$$\frac{1}{T_2^*} = \frac{1}{T_2} + \frac{\gamma}{2\pi} \cdot \frac{\Delta B_0}{2}$$

2.34)

where ΔB_0 is the magnetic field inhomogeneity. (Note that $T_2^* \leq T_2$). Unless T_2 is much less than $4\pi/(\gamma \Delta B_0)$, the decay time T_2^* cannot be used as a measure of transverse relaxation.

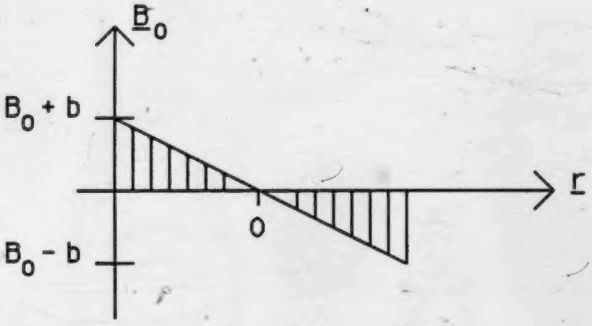


Figure 2.7 The purposeful reduction of $\underline{B}_0$ by a linear magnetic field gradient. b is the additional magnetic field component at some location along a direction given by $\underline{r}$.

Hahn (1950) proposed a method, which he called the spin echo method, that overcomes this inhomogeneity problem. One variation of this method (Meiboom and Gill, 1958) can be written as $90°_{+x}$-τ-$180°_{+y}$-τ-echo, which reads as a 90 degree RF pulse along the x_ρ axis (ρ stands for rotating frame) followed by a delay τ, then a 180 degree pulse along the y_ρ axis followed by a second delay of τ, and then the spin echo (Figure 2.8).

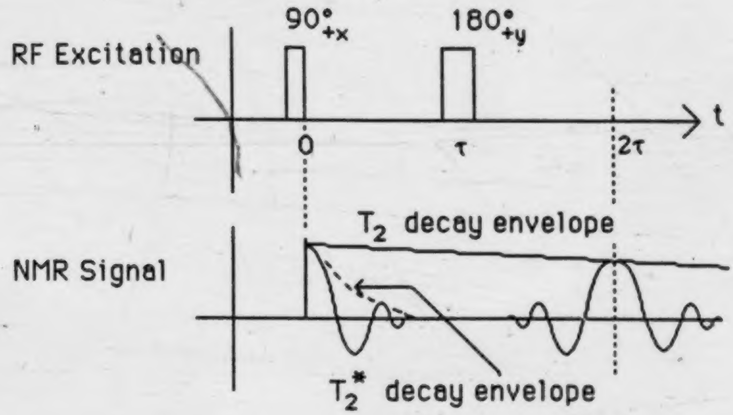


Figure 2.8 A spin echo pulse sequence and the detected signal.

During the first τ delay the spins dephase as illustrated in Figure 2.6a). The application of a 180 degree pulse along the y_p axis rotates the spins 180 degrees about y_p (Figure 2.9a). the spins continue to process in the same direction and will, after another τ seconds, refocus along the y_p axis (Figures 2.9b and 2.9c). The rotation of these alternately dephasing and rephasing transverse magnetization components in the laboratory frame results in an induced EMF. This signal will decay for τ seconds after the initial 90 degree pulse and then regrow to a maximum, or spin echo, τ seconds after the 180 degree pulse. Further spin echoes can be generated by applying additional 180 degree pulses, each separated by 2 τ. A limit to the number of spin echoes that can be induced in this manner is imposed by the irreversible, or true T_2 decay process. The amplitudes of successive echoes will decrease according to true transverse relaxation. Thus, T_2 can be determined from a plot of echo amplitude as a function of τ.

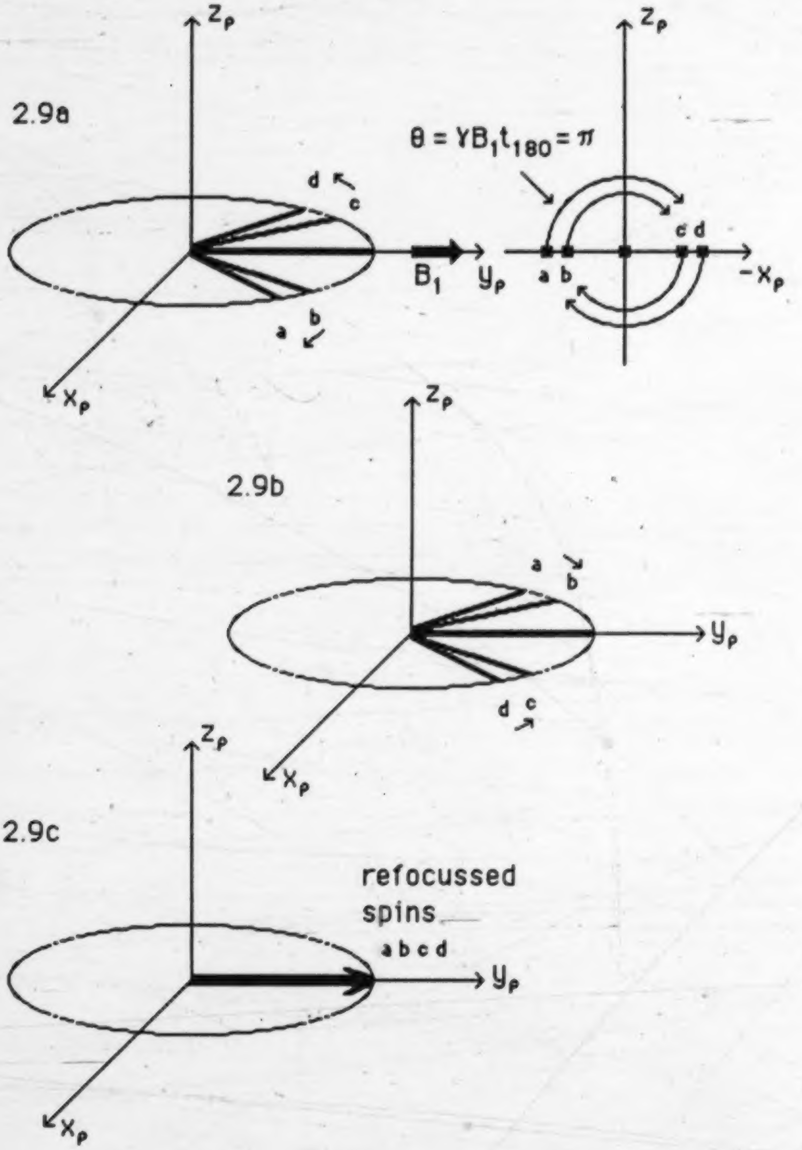

2.9a
zρ
xρ
θ = γB1t180 = π
d
c
b
a
B1
yρ
a b
c d
−xρ
2.9b
zρ
a
b
yρ
d
c
xρ
2.9c
zρ
refocussed
spins
a b c d
yρ
xρ

Figure 2.9 a) The rotation of the components of M_{xy} about the y_ρ axis by the 180°_{+y} pulse. b) The rephasing of the components during the second delay τ. c) The completely rephased transverse magnetization.

This measurement is accurate as long as the sample is of homogeneous composition and there is no molecular diffusion. A heterogeneous sample will give a T_2 value that is a weighted sum of the T_2 values of the individual molecular species. Molecular diffusion causes nuclei comprising the transverse magnetization to move from one part of the inhomogeneous magnetic field to another, and hence causes additional dephasing. As diffusion is a random process, the dephasing due to this process cannot be reversed. Thus the amplitudes of the echoes will be reduced.

2.8 Two Dimensional Fourier Transform Imaging

The objectives in any NMR imaging technique are to measure NMR parameters, such as the NMR visible spin density and T_1 or T_2, as a function of their spatial coordinates. In a heterogeneous sample these parameters vary within the sample volume. Since the only signal available is that arising from the precession of the magnetization in the sample, the way to measure the NMR parameters at different spatial locations is to uniquely specify the rate of magnetization precession for a given location. This is done by superimposing a static linear gradient onto the static magnetic field so that the magnetization at different parts of the sample precesses at different frequencies. Through this, the spatial displacements in the sample are related to frequency displacements. The linear gradients modify the strength of the static magnetic field, $\underline{B}_0$, along a given direction. The field experienced by the magnetization in isochromatic planes normal to the gradient direction is given by

$$\underline{B}' = \underline{B}_0 + (\underline{G}_r \cdot \underline{r})\underline{k} \qquad 2.35)$$

The magnetic field gradient, $\underline{G}_r$, has the components

$$G_x = \partial B_z/\partial x,\ G_y = \partial B_z/\partial y,\ G_z = \partial B_z/\partial z \qquad 2.36)$$

Thus,

$$\underline{G}_r = (G_x\underline{i} + G_y\underline{j} + G_z\underline{k}) \qquad 2.37)$$

The position vector $\underline{r}$ has its origin at the centre or zero point of the gradient (where only $\underline{B}_0$ is present). The dot product in equation 2.35) gives, for a specific $\underline{r}$ value, the contribution to $\underline{B}_0$ in a plane normal to $\underline{r}$ (i.e., a plane where $\underline{G}_r \cdot \underline{r}$ equals a constant). In this plane, magnetization components normal to $\underline{B}_0$ will precess at

$$\omega(\underline{r}) = \gamma(B_0 + \underline{G}_r \cdot \underline{r}) \qquad 2.38)$$

For example, if, after a 90 degree pulse has been applied to the whole sample, a linear gradient is established in the direction of the laboratory frame x axis, then the FID acquired will be comprised of sinusoids at each of the resonant frequencies defined by the gradient within the sample. If the acquired signal is Fourier transformed, a projection of the NMR visible spin density for each of the isochromatic planes normal to the gradient direction onto that direction results (Figure 2.10).

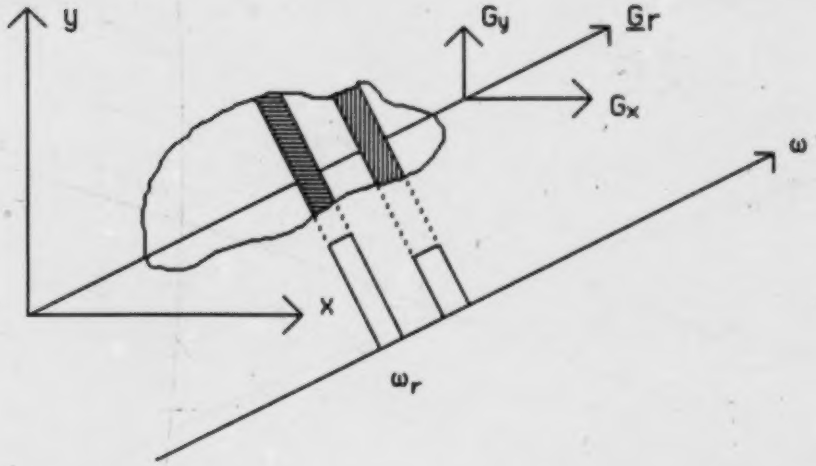


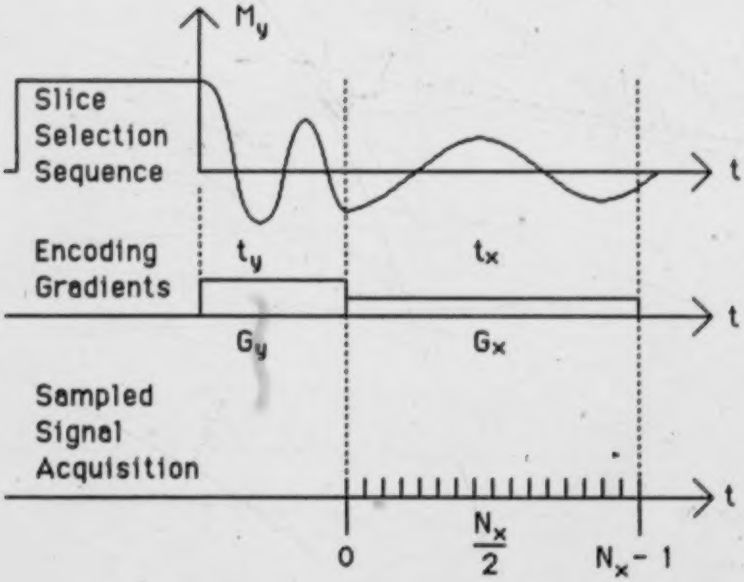
Figure 2.10 The projection of two arbitrary spin distributions in a sample in a magnetic field gradient $\underline{G}_r$ onto a frequency axis parallel to $\underline{G}_r$.

Various imaging techniques, such as Image Reconstruction by Projection (IRP) and Echo-planar imaging, are based on this " projection " scheme. As indicated by the title of this section, this thesis is concerned with Two Dimensional Fourier Transform (2DFT) image reconstruction as proposed by Kumar, Welti and Ernst (1975).

Since the images that are generated are two dimensional representations of finite sections or slices through the sample, the logical starting point in an imaging discussion would be with an explanation of slice selection. However, for the time being, assume that a slice has already been defined in which the magnetization lies in the transverse plane (normal to $\underline{B}_0$). It will be easier to backtrack later and explain the slice definition process. A consequence of a finite slice thickness is that, although images are shown as two dimensional arrays of pixels, each pixel actually represents a volume element, or voxel. The voxel volume is given by the area of the pixel in the image multiplied by the slice thickness.

For the following discussion the rotating frame of reference will be used unless otherwise noted. Axes referring to the rotating frame will have the subscript ρ.

After the slice defining sequence, the magnetization in the slice will be parallel to the y_ρ axis. During the resultant FID (detected in the laboratory frame of reference), two orthogonal linear magnetic field gradients (henceforth known as gradients) are applied sequentially as illustrated in Figure 2.11.



M_y
Slice Selection Sequence
t
Encoding Gradients
t_y
t_x
t
G_y
G_x
Sampled Signal Acquisition
t
0
$\frac{N_x}{2}$
$N_x - 1$

Figure 2.11 A possible sequence of gradient pulses used in 2DFT imaging.

The gradient " pulse ", G_y, encodes the orthogonal cartesian coordinate , y, into the NMR signal by governing the phase of the signal at the start of the next gradient pulse, G_x. To illustrate this encoding (and following arguments), consider two elemental magnetizations, each of magnitude m_0, placed in the laboratory frame of reference x-y plane as seen in Figure 2.12.

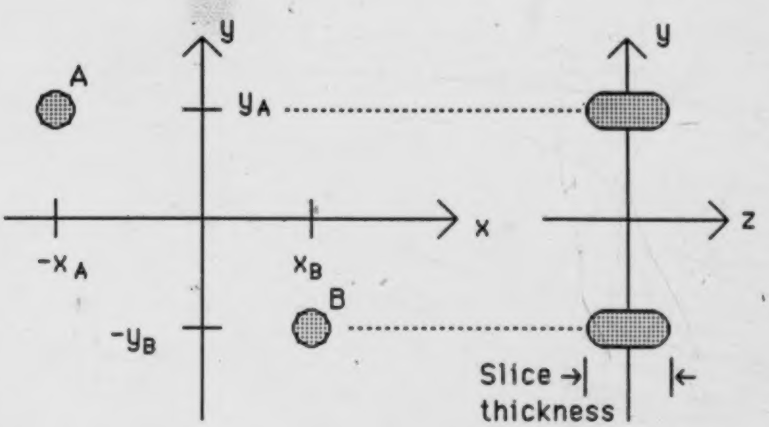


Figure 2.12 Two elemental magnetizations.

Under the influence of G_y the precessional frequencies for the elemental magnetizations in the laboratory frame are given by

$$\omega_A = \gamma (B_0 + G_y y_A) \quad 2.39)$$
$$\omega_B = \gamma (B_0 - G_y y_B) \quad 2.40)$$

In the rotating frame

$$\omega_{A\rho} = \gamma G_y y_A \quad 2.41)$$
$$\omega_{B\rho} = - \gamma G_y y_B \quad 2.42)$$

The angle between the magnetizations and the y_p axis at the end of the G_y encoding period would be

$$\theta_A = \gamma G_y y_A t_y \tag{2.43}$$

$$\theta_B = -\gamma G_y y_B t_y \tag{2.44}$$

The components of the magnetizations in the rotating frame after the time t_y are

$$m_{y\,A} = m_0 \cos\theta_A \tag{2.45}$$

$$m_{x\,A} = m_0 \sin\theta_A \tag{2.46}$$

$$m_{y\,B} = m_0 \cos\theta_B \tag{2.47}$$

$$m_{x\,B} = m_0 \sin\theta_B \tag{2.28}$$

This is referred to as the " phase encoding " step in the sequence. After t_y seconds the phase encoding gradient, G_y, is switched off and replaced by the G_x gradient. The purpose of G_x is to encode the x spatial cartesian coordinate into the NMR signal. It governs the elemental magnetization rotation frequencies and hence, the frequencies of the sinusoids of the detected signal. This signal, acquired using discrete samples during the time t_x, is Fourier transformed and, as mentioned before, the resulting frequency components can be related to the distribution of spins in isochromatic planes normal to, in this case, the x direction.

Returning to the elemental magnetizations (Figure 2.12), it can be seen that under the influence of G_x, the rotating frame components of the magnetizations become

$$m_{y\,A} = m_0\cos[\,\theta_A - \gamma G_x x_A(t - t_y)\,] \qquad 2.49)$$

$$m_{x\,A} = m_0\sin[\,\theta_A - \gamma G_x x_A(t - t_y)\,] \qquad 2.50)$$

$$m_{y\,B} = m_0\cos[\,\theta_B + \gamma G_x x_B(t - t_y)\,] \qquad 2.51)$$

$$m_{x\,B} = m_0\sin[\,\theta_B + \gamma G_x x_B(t - t_y)\,] \qquad 2.52)$$

Quadrature Phase Sensitive Detection yields both components of the signal from each elemental magnetization. Since the recorded signal is shifted to be centred about $\omega = 0$, it is thought of as a signal detected in the rotating frame of reference. That is,

$$x_{p\ \mathrm{component}} = m_{x\,A} + m_{x\,B} = \mathbf{F}_R$$

$$y_{p\ \mathrm{component}} = m_{y\,A} + m_{y\,B} = \mathbf{F}_I$$

where $\mathbf{F}_R$ is defined as the real part of the NMR signal and $\mathbf{F}_I$ as the imaginary part. The total signal is given by $\mathbf{F} = \mathbf{F}_R + i\mathbf{F}_I$. The complex Fourier transform of this signal yields, after some algebraic manipulations, terms such as

$$m_{0\xi}\cos\theta_\xi\,\delta(\,\omega \pm \gamma G_x x_\xi\,) \qquad 2.53)$$

$$im_{0\xi}\sin\theta_\xi\,\delta(\,\omega \pm \gamma G_x x_\xi\,) \qquad 2.54)$$

where ξ is either A or B (the elemental magnetization index) and $\tau = (t\text{-}t_y)$. The real part of the frequency spectrum of the signal from the two elemental magnetizations is shown in Figure 2.13. In the real part of the frequency spectrum, the magnitude of each component is governed by the $\cos\theta_\xi$ term in equation 2.53). Likewise, the magnitudes of the components in the imaginary part of the spectrum are governed by the $\sin\theta_\xi$ term in equation 2.54).

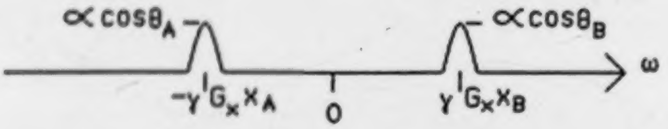


Figure 2.13 The real part of the frequency spectrum from one phase encoded scan. The component amplitudes are proportional to the $\cos\theta_\xi$ term in equation 2.53).

This is a projection of complex values for the spin densities of each elemental magnetization, as governed by the phase encoding gradient, onto the frequency encoding direction axis, in this case the x axis. In order to complete the two dimensional NMR data set, a sequence of FIDs must be acquired which will specify the position of the magnetization in the phase encoding or y direction. This is accomplished by varying either the time, t_y, or the gradient strength, G_y, for each FID generated. This will give different phase angles, as defined in equations 2.43) and 2.44), for each FID. The result, after each FID has been Fourier transformed, will be a complex data matrix, the real part of which is shown in Figure 2.14.

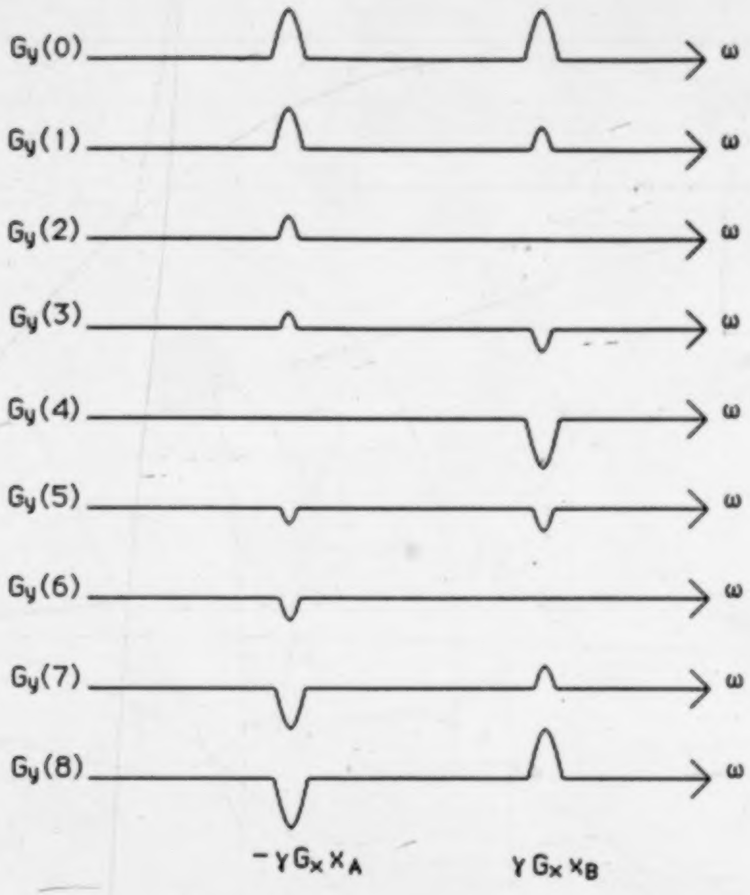

$G_y(0)$
$G_y(1)$
$G_y(2)$
$G_y(3)$
$G_y(4)$
$G_y(5)$
$G_y(6)$
$G_y(7)$
$G_y(8)$
ω
$-\gamma G_x x_A$
$\gamma G_x x_B$

Figure 2.14 An array of the real part of the frequency domain data following the first Fourier transformation of the acquired time domain data. The rows are indexed by the value of the phase encoding gradient $G_y(i)$ where
$G_y(i) = (i-N_y/2)\Delta G$; $i = 0, 1, 2, \ldots, N_y-1$.

In this matrix (Figure 2.14) the x coordinate of each elemental magnetization is clearly specified. It can be seen that the amplitudes of the signal at x coordinates vary sinusoidally from one row or scan to the next.

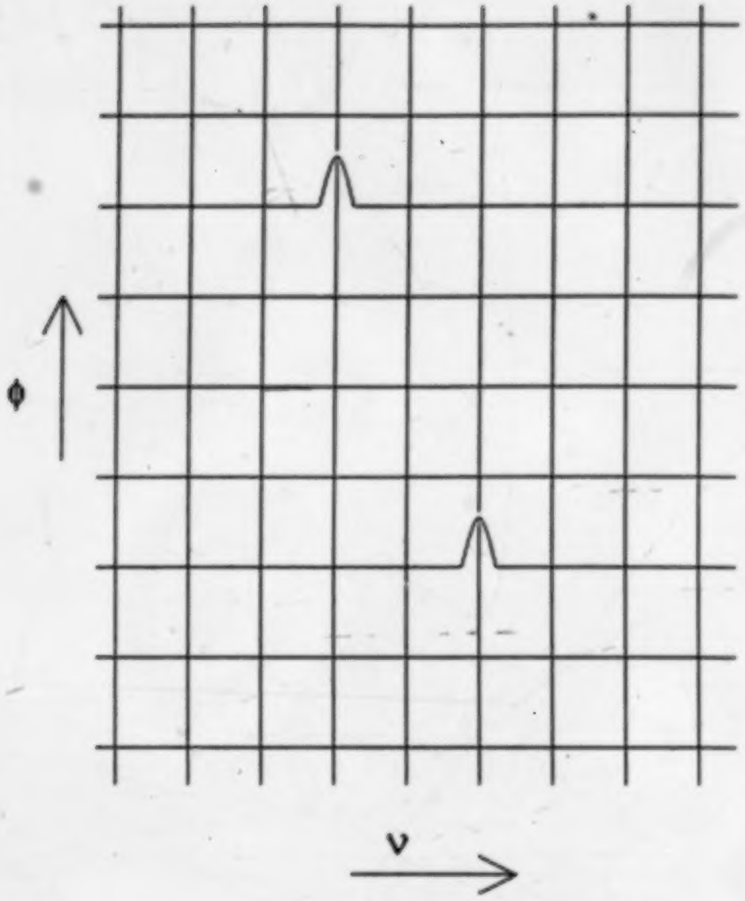


Figure 2.15 The real part of the two dimensional complex Fourier transform of the original time domain data set. Converting the signal amplitudes into pixel intensities gives an image of the two elemental magnetizations. In this figure, ϕ stands for the phase encoding direction and ν for the frequency encoding direction.

The frequency of the sinusoidal change in the amplitudes of the finite components at each x coordinate is dependent on the strength of the phase encoding gradient pulse in time at the y location corresponding to each elemental magnetization. That is, the frequency in the phase encoding direction depends upon how rapidly the term θ_ξ (equations 2.53) and 2.54)) changes from scan to scan. If the data are complex Fourier transformed in the vertical direction, the position of the magnetizations in the phase encoding or y direction is specifed (remember the amount of phase encoded is proportional to the location of the magnetization in the phase encoding gradient). In this Fourier transform in the vertical direction, the real part of each scan line spectrum in Figure 2.14 will give the frequency of each elemental magnetization along the phase encoding direction and the imaginary part of the scan line spectrum will indicate the sign of the frequency. Thus the locations of the elemental magnetizations will be specified in the phase encoding direction. The final result is a two dimensional complex data matrix representing the complex two dimensional frequency spectrum of the data acquired over N_y scans. The non-zero components of this matrix can be related to the spatial locations of the elemental magnetizations by using equations 2.41) through 2.44). Images can be generated from this matrix that represent the modulus of the spectral components ($[\, m_{x\xi}^2 + m_{y\xi}^2 \,]^{1/2}$), or that represent the real or imaginary components ($m_{x\xi}$ or $m_{y\xi}$), or that represent the phase ($\theta = \arctan[\, m_{y\xi}/m_{x\xi} \,]$). This is done by taking the components or calculated quantities desired and converting them to pixel intensities.

2.9 Slice Definition

This section returns to the problem of defining the plane or slice within the sample from which the image is to be obtained. The assumption in section 2.8 was that the magnetization within such a slice was rotated by a 90 degree RF pulse onto the y axis of the rotating frame and that the magnetization outside the slice was left undisturbed, i.e., in thermal equilibrium.

To accomplish this a linear magnetic field gradient is applied in the direction normal to the desired slice. The magnetization in isochromatic planes normal to the direction of this gradient will experience the same magnetic field. Hence, the spins in the slice of interest will have a unique Larmor frequency with respect to spins elsewhere in the sample. Irradiating the sample only at the frequency specific to that slice will rotate the spins in that slice only. However, application of a radiofrequency electromagnetic radiation pulse at one frequency, or even a narrow band of frequnecies, is not simple.

It has been shown that the approximate shape of the slice can be obtained through Fourier transform analysis of the time domain RF pulse (Locher, 1980). This analysis is approximate because the NMR interaction is non-linear. That is, linear signals input to the system (the pulse sequence) elicit a non-linear output. Fourier transform analysis of the input and output signals of a system presupposes a linear relationship between the two signals (i.e., a linear transfer function). Thus, such an analysis of a NMR experiment, which is non-linear, can be only, at best, approximate. A better approach considers the behaviour of the magnetization components in the sample under the irradiating sequence (Mansfield et al., 1979; Locher, 1980).

In section 2.3, which dealt with the effect of alternating magnefic fields, it was seen that in the rotating frame the magnetization will precess about an effective magnetic field, $\underline{B}_{eff}$, comprising the RF time varying magnetic field, $\underline{B}_1$, and the static magnetic component in the direction of $\underline{B}_0$. Remember that when the RF field was exactly on resonance, $\underline{B}_{eff}$ was simply $\underline{B}_1$ and the magnetization precessed about $\underline{B}_1$. To account for a magnetic field gradient, equation 2.22) must be modified to

$$\underline{B}_{eff} = \underline{k}(B_0 + \underline{G}_r \cdot \underline{r} + \omega/\gamma) + \underline{i}B_1 \qquad 2.55)$$

where $\underline{G}_r$ is the magnetic field gradient vector along the position vector $\underline{r}$. In this case r is thought of as the distance from the centre of the gradient (where only the field $\underline{B}_0$ is present) along the gradient direction to the centre of the desired slice. Hence, to satisfy the resonance condition at the centre of the slice, ω must be equal to $\gamma(B_0 + \underline{G}_r \cdot \underline{r})$. Conversely, ω may be set to γB_0, the frequency of the rotating frame of reference, in which case $| \underline{B}_{eff}(\underline{r}) |$ is $[B_1^2 + (\underline{G}_r \cdot \underline{r})^2]^{1/2}$ (see Figure 2.16).

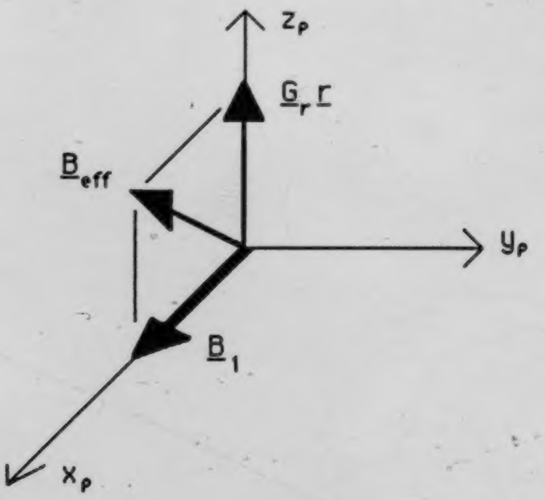


Figure 2.16 Effective magnetic field in the rotating frame if $\omega = \omega_0 = \gamma B_0$.

It's easy to see that if $B_1 >> |\underline{G}_r \cdot \underline{r}|$ for all $\underline{r}$, then $\underline{B}_{eff}(\underline{r}) \approx \underline{B}_1$ for all $\underline{r}$. This means that all the spins in the sample will experience the same $\underline{B}_{eff}$ and will, for a given RF pulse, rotate the same amount in the y-z plane. Thus, all the spins within the sample will be equally excited. Conversely, if $0 < B_1 << |\underline{G}_r \cdot \underline{r}|_{r>>0}$, then $\underline{B}_{eff}(\underline{r})$ will vary markedly along the direction of $\underline{r}$. Only those spins in planes near r = 0 will experience $\underline{B}_{eff}(\underline{r}) = \underline{B}_1$. These spins will interact strongly with $\underline{B}_{eff}$ and, for a 90 degree pulse, be rotated into the x-y plane. Spins far from r = 0 will experience a $\underline{B}_{eff}$ that is almost parallel to $\underline{B}_0$ and will not rotate from $\underline{B}_0$ at all. The intermediate case of $B_1 \approx |\underline{G}_r \cdot \underline{r}|$ gives a $\underline{B}_{eff}$ that is parallel neither to the z nor to the x axis. The magnetization in planes where this is true will have components along all three axes. Such a case is often termed an " off-resonance " effect, i.e., the reacting spins away from r = 0 are " off resonance " with respect to the frequency of irradiation. Figure 2.17 illustrates the above discussed effects as seen after a rectangularly modulated 90 degree RF pulse.

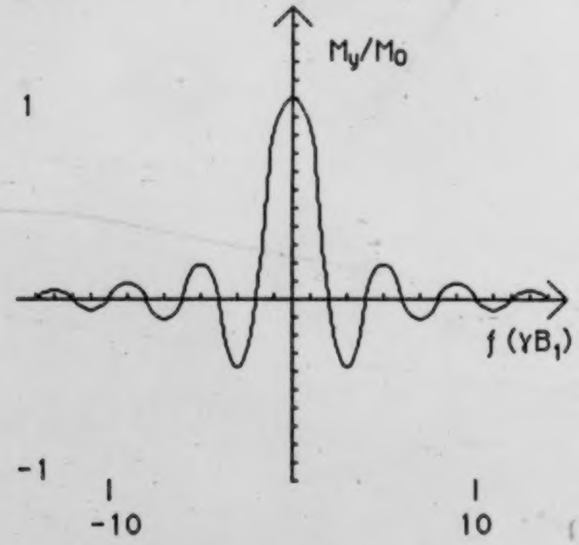


Figure 2.17

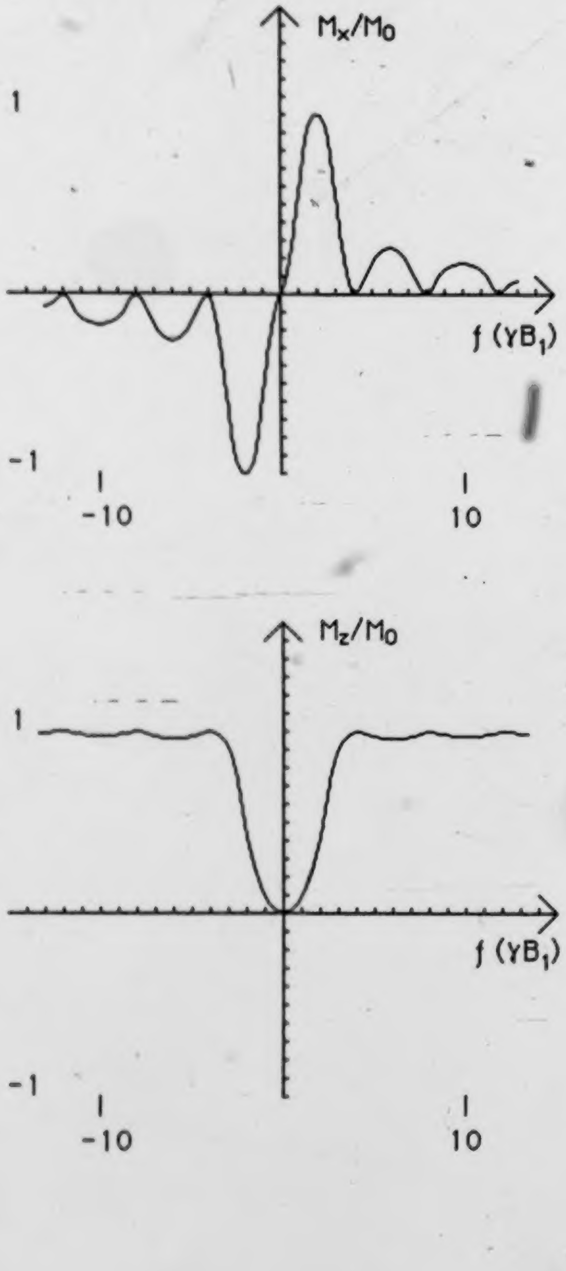


Figure 2.17

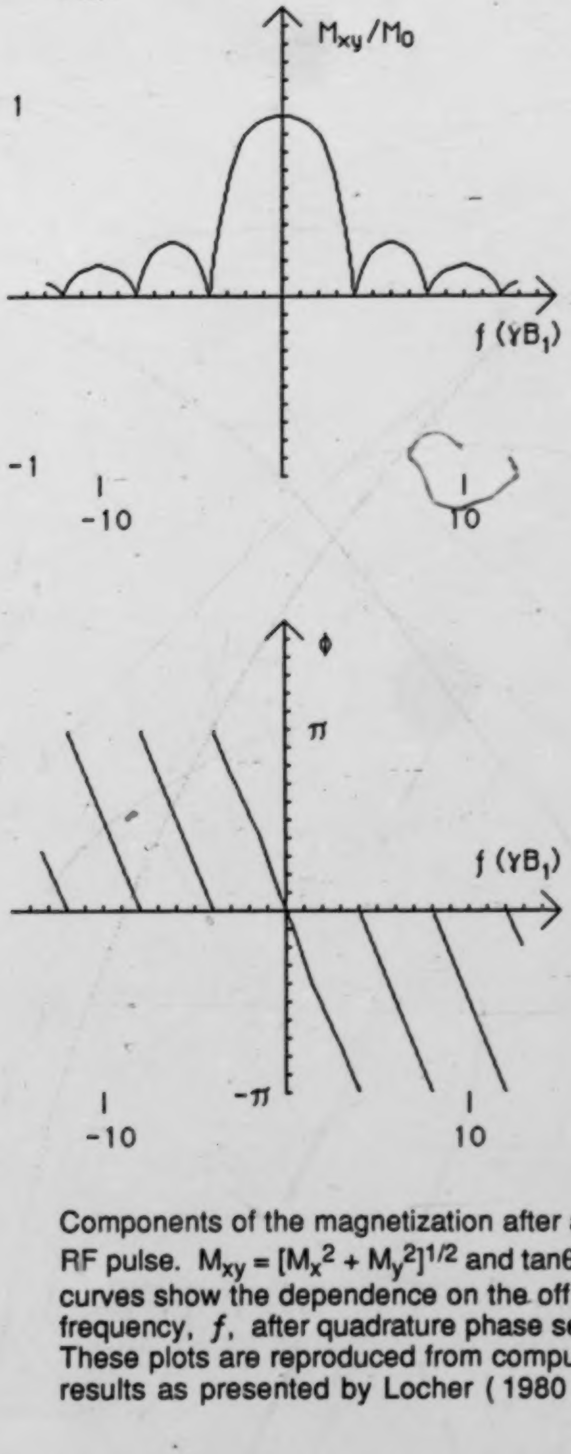


Figure 2.17 Components of the magnetization after a rectangular 90° RF pulse. $M_{xy} = [M_x^2 + M_y^2]^{1/2}$ and $\tan\theta = -M_x/M_y$. The curves show the dependence on the off resonance frequency, f, after quadrature phase sensitive detection. These plots are reproduced from computer simulation results as presented by Locher (1980).

Notice that the initial phase of the magnetization components in the x-y plane is distributed over 2π. The signal resulting from the vector sum of these components would be much smaller than the magnitude summation of these components. Notice also that the central lobe of the M_{xy}/M_0 plot, defining the slice in the sample, is not sharply defined and that there are appreciable M_{xy} components outside the slice that would contribute to the observed signal. It is clear that these phase distributions and " side lobes " must be corrected.

The resemblance of the M_y/M_0 plot to the Fourier transform of a rectangular time domain pulse led to the speculation that a more suitable time domain pulse shape would be the Fourier transform of a square distribution of frequency components in the frequency domain. The Fourier transform of such a distribution yields a time domain sin(x)/x, or sinc(x) pulse. Since such a pulse is infinite in time, it must be windowed by some other function. This function is usually chosen so as to maintain as square a distribution of frequency domain components as possible. Locher (1980) showed the results of computer simulations for a Gaussian windowed sinc-shaped RF pulse. The result, for a RF pulse less than or equal to 90 degrees, was a much better defined slice profile with reduced side lobes. The phase of the transverse magnetization components was still distributed over 2π. The application of a gradient pulse in the opposite direction and of the same magnitude for a time equal to half of the 90 degree pulse duration following the shaped 90 degree RF pulse (Figure 2.18) would rephase these components (Sutherland and Hutchison, 1975). The result would be a large net magnetization vector along the y axis of the rotating frame. However, due to the aforesaid interaction of the effective magnetic field with off-resonance spins, pulse angles greater than 90 degrees resulted in non-uniform distributions of spin excitation in the slice and spin excitation away from the slice. Another suitable time domain RF pulse shape, for pulses of 90 degrees and less, is a Gaussian envelope (Sutherland and Hutchison, 1975).

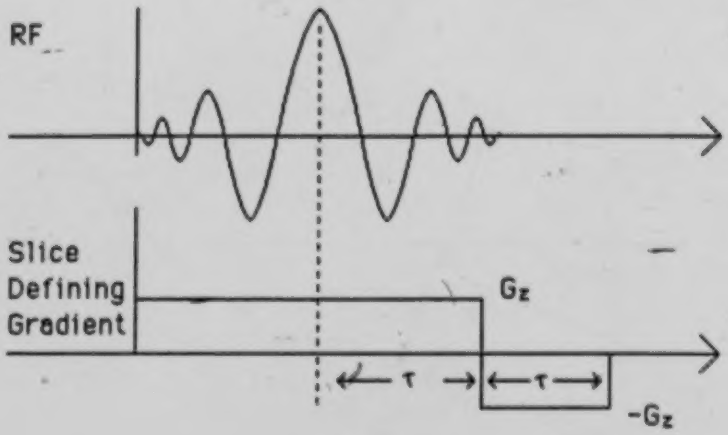


Figure 2.18 A slice defining sequence.

2.10 A Mathematical Development of the Two Dimensional Fourier Transform Imaging Technique

To complete this treatment of 2DFT NMR imaging, the mathematical equation describing the 2DFT reconstruction as given by Wood and Henkelman (1985) is presented.

The imaging sequence they present, which is the basic sequence for the flow imaging technique presented later, utilizes spin echoes (Figure 2.19). The analysis of the gradient and RF pulse sequence is the same as that presented by Kumar et al. (1975) for a gradient modulated

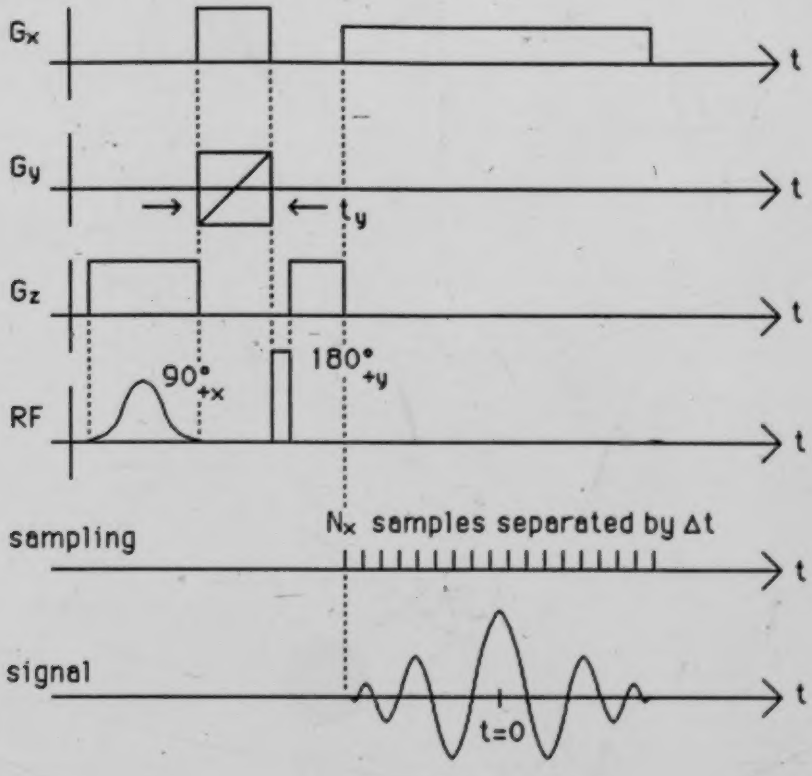

G_x
G_y
G_z
RF
sampling
signal
t
t
t
t
t
t
t_y
90°_{+x}
180°_{+y}
N_x samples separated by Δt
t=0

Figure 2.19 Gradient and RF pulse timing diagram.

FID. The spin echo may be considered as a time-reversed FID followed immediately by another FID, forward in time. The only difference, as far as the mathematical analysis is concerned, between the gradient modulated FID technique and that using spin echoes is the temporal location of the phase encoding gradient. A change from the sequence as presented by Wood and Henkelman, but one that does not affect this analysis, is in the temporal position of the slice " re-phasing " gradient pulse. The rephasing gradient pulse is applied after the 180 degree RF pulse. It is the same amplitude and in the same direction as the gradient pulse applied during the

90 degree RF pulse, and is half the duration of the 90 degree RF pulse. This slice selection scheme is equivalent to that illustrated in Figure 2.18 since the 180 degree RF pulse acts to invert the sign of the phase of the transverse magnetization. For example, if a component of the transverse magnetization had a phase of $+\xi$ (with respect to the y_ρ axis) sometime after a 90 degree RF pulse applied along x_ρ, it would have a phase of $-\xi$ immediately following a 180 degree RF pulse along y_ρ.

Figure 2.19 shows the pulse sequence used to acquire one phase encoded signal. In this case the amplitude of the phase encoding gradient pulse is varied from one signal acquisition to the next. Each echo is sampled N_x times at a rate of $1/\Delta t$. The sequence is repeated N_y times with a delay of T_R seconds between each acqusition. It is assumed that T_R is long enough to ensure that complete recovery to equilibrium occurs. Therefore, the effect of longitudinal relaxation is ignored in this analysis. The phase encoding, or y gradient is incremented monotonically N_y times in steps of ΔG, starting from $[\,1 - N_y/2\,]\Delta G$. In practical implementations, the phase of the RF pulses (i.e., the orientation of the alternating magnetic field vector, $\underline{B}_1$, in the rotating frame) is manipulated to suppress zero offsets and to cancel FIDs following non-selective 180 degree RF pulses. In this analysis it is assumed that only one signal is recorded per phase encoding gradient step. The magnetization at any given (x,y) location in the sample is modelled as a delta function which emits a sinusoidal signal. The spin echo is the sum of the sinusoidal signals emitted by the isochromatic magnetizations in the slice modified by an apodization function, $E_x(t)$. The narrow frequency bandwidth 90 degree RF magnetic field pulse (illustrated in Figure 2.19), in the presence of a slice defining gradient along the z axis, rotates the magnetization in a narrow slice by 90 degrees onto the y_ρ axis. The 180 degree pulse is non-selective, i.e., it rotates all the magnetizations in the sample by 180 degrees. Magnetization outside the slice remains either parallel or anti-parallel to the main field, $\underline{B}_0$, throughout the experiment and, thus, does not contribute to the signal acquired. The amplitude of the first G_x gradient pulse is adjusted so that the centre of the echo occurs at sample point $N_x/2$. This is defined as the time origin for the analysis.

The phase of the signal for a magnetization at position (x,y) is then given as

$$\theta(t,G_y) = \gamma\,(\,G_x x\,t \;-\; G_y y\,t_y\,) \tag*{2.56)}$$

The signal is detected using Quadrature Phase Sensitive Detection at a reference frequency equal to the Larmor frequency ($\omega_0 = \gamma B_0$). The data set for one image is composed of N_y complex spin echo signals, each with different amounts of phase encoding and modified by another apodization function in the y direction, $E_y(G_y)$. The complete expression is

$$\begin{aligned} s(t,G_y) = {} & \{\, M_0\, E_x(t)\, E_y(G_y) \exp[\, i\theta(t,G_y)\,]\, d(t)\, d(G_y)\,\} \\ & \otimes_t \Uparrow [\, t/(N_x\Delta t)\,] \otimes_{Gy} \Uparrow [\, G_y/(N_y\Delta G)\,] \end{aligned} \tag*{2.57)}$$

where

$$d(t) = \Uparrow [\, t/\Delta t\,] \;\; \mathrm{rect}\,[\, t/(N_x\Delta t)\,] \tag*{2.58)}$$

$$d(G_y) = \Uparrow [\, G_y/\Delta G\,] \;\; \mathrm{rect}\,[\, G_y/(N_y\Delta G)\,] \tag*{2.59)}$$

$$\Uparrow\left[\frac{x}{\Delta x}\right] = \Delta x \sum_{k=-\infty}^{\infty} \delta(x - k\Delta x) \qquad \text{the sampling function}$$

2.60)

$$\text{rect}\left[\frac{x}{N\Delta x}\right] = \begin{cases} 0 & x \leq \frac{-N\Delta x}{2}, \\ 1 & \frac{-N\Delta x}{2} < x \leq \frac{N\Delta x}{2}, \\ 0 & \frac{N\Delta x}{2} < x \end{cases}$$

2.61)

and M_0 is the magnitude of the transverse magnetization at $t = 0$. Convolution is denoted by $\otimes$ and the domain is indicated by a subscript.

Using the fact that the Fourier transform kernel is separable and convolution in the time domain is the same as multiplication in the frequency domain, and vice versa, two one-dimensional Fourier transforms with respect to the variables t and G_y in equation 2.57) may be taken, yielding

$$\begin{aligned} \mathbb{F}\{\mathbb{F}[s(t,G_y)] \} = {} & [\mathbb{F}(\mathbb{F}\{ M_0 \exp[i\theta(t,G_y)] \}) \\ & \otimes_{fx}\mathbb{F}[E_x(t)] \otimes_{fy}\mathbb{F}[E_y(G_y)] \\ & \otimes_{fx}\mathbb{F}[d(t)] \otimes_{fy}\mathbb{F}[d(G_y)]] \\ & * \Uparrow (N_x \Delta t\, f_x) \Uparrow (N_y \Delta G\, f_y) \qquad 2.62) \end{aligned}$$

where $\mathbb{F}$ is defined as the Fourier transform operator and

$$f_x = (\gamma/2\pi)\, G_x x \qquad 2.63)$$

$$f_y = -(\gamma/2\pi)\, t_y y \qquad 2.64)$$

Since the signal originates from a delta function at (x_0, y_0), the phase term in equation 2.56) is simple and gives for the first factor in equation 2.62)

$$\mathbb{F}(\mathbb{F}\{ M_0 \exp[\, i\theta(t,G_y)] \}) = M_0\, \delta[\, f_x - (\gamma/2\pi)G_x x_0] \, \delta[\, f_y + (\gamma/2\pi)t_y y_0] \qquad 2.65)$$

Remembering that the spin echo is considered to be a time-reversed FID followed by a mirrored FID, $E_x(t)$ is seen to be a decaying exponential envelope. Thus, $\mathbb{F}[\, E_x(t)]$ is a Lorentzian envelope. Wood and Henkelman (1985) state that there is no apodization in the G_y direction. There is no apodization since it is assumed that return to thermal equilibrium occurs between each scan. Thus $E_y(G_y)$ is always equal to one and $\mathbb{F}[\, E_y(G_y)]$ is a delta function at $f_y = 0$. The multiplication of d(t) and $d(G_y)$ by a rectangle in the time domain is the same as the convolution of the Fourier transforms of these terms with a sinc function in the frequency domain. That is,

$$\mathbb{F}[\, d(t)] = N_x^2\, (\, \Delta t\,)^3 \Uparrow (\, \Delta t\, f_x\,) \otimes_{f_x} \mathrm{sinc}(\, N_x\, \Delta t\, f_x\,) \qquad 2.66)$$

where sinc(x) is defined as $\sin(\pi x)/(\pi x)$. Applying equations 2.63) and 2.64) to the Fourier transformed signal in equation 2.62) provides the point spread function

$$\begin{aligned} PSF(x,y) = K\,[\,& P(x,y) \otimes_x \mathbb{F}[\,E_x(t)\,] \otimes_y \mathbb{F}[\,E_y(G_y)\,] \\ & \otimes_x D(x) \otimes_y D(y)\,]^{*} \\ & \Uparrow[\,N_x\,\Delta t\,(\gamma/2\pi)\,G_x\,x\,]^{*} \\ & \Uparrow[\,N_y\,t_y\,\Delta G\,(\gamma/2\pi)\,y\,] \end{aligned} \qquad 2.67)$$

where

$$P(x,y) = M_0\,\delta[\,x - x_0\,]\,\delta[\,y - y_0\,] \qquad 2.68)$$

$$\begin{aligned} D(x) = & \Uparrow[\,\Delta t\,(\gamma/2\pi)\,G_x\,x\,] \\ & \otimes_x \mathrm{sinc}[\,N_x\,\Delta t\,(\gamma/2\pi)\,G_x\,x\,] \end{aligned} \qquad 2.69)$$

$$\begin{aligned} D(y) = & \Uparrow[\,\Delta G\,(\gamma/2\pi)\,t_y\,y\,] \\ & \otimes_y \mathrm{sinc}[\,N_y\,t_y\,\Delta G\,(\gamma/2\pi)\,y\,] \end{aligned} \qquad 2.70)$$

K represents the abitrary constants in the equation. Note that P(x,y) represents the transverse magnetization at a point (x,y) while all the other terms broaden the distribution. These distributions are a function of the pulse sequence and, for a given set of pulse sequence parameters, are sample independent (that is, neglecting such effects as the magnetic susceptibility of the sample, inhomogeneities, electronic drift and noise, etc.).

3. Flow Measurements Using NMR Imaging

Blood flow in the vascular network of humans is of considerable interest for medical diagnosis and has been the subject of research since William Harvey (1578 - 1657) published De Motu Cordis in 1628 (McDonald, 1974). More recent research has made use of NMR imaging techniques to quantify blood flow parameters (Axel, 1984).

Initial efforts were based on the effects observed in modulus images of blood flowing in vessels within the slice and intersecting the slice. The details of these effects depend on the imaging sequence used and on the magnitude of the flow. For example, when a spin echo sequence using selective excitation and selective refocussing pulses is used, the amplitude of the spin echo will first increase with increasing flow rate due to the replacement of partially saturated spins by more strongly magnetized spins from upstream of the imaging volume. The degree of saturation, and hence the degree of signal gain, is dependent on the repetition time and the rate of longitudinal relaxation. At higher flow velocities there is a decrease in the signal because the spins " washout " of the selective refocussing region, i.e., not all of the initially excited spins are still in the slice when the selective 180 degree RF pulse is applied. The signal also decreases due to the dephasing effects of motion along imaging magnetic field gradients. (Arnold and Burkhart, 1965; Crooks et al., 1984). Axel (1984) demonstrated spin echo amplitude flow measurement techniques which used selective and non-selective refocussing pulses. He correlated modulus image pixel amplitudes with flow velocity using arguments similar to those above.

A different approach, known as the time-of-flight method, measures flow velocity by means of the measurement of the distance travelled by a group of spins in a known time. One such method observes the change in modulus images due to flow in images downstream from a selected image in a multi-slice imaging sequence (Bradley et al., 1984). A second method, used by Deimling et al. (1986), applys a selective refocussing pulse at a known distance downstream from the selective excitation pulse location. The signal detected depends on the volume of selected spins that flows into the refocussing area or slice in the time between the excitation and refocussing pulses. This information is used to calculate the flow velocity.

Other researchers have approached fluid flow velocity characterization by examining how the phase of the recorded signal is affected by flow. As stated before, the signal from an elemental magnetization moving along the direction of an applied magnetic field gradient acquires a phase shift with respect to the signal from non-moving magnetizations. Van Dijk (1984), Bryant et al. (1984) and O'Donnell (1985) have used pulsed gradient sequences which do not affect the phase of the signal from stationary magnetizations, but that alter the phase of the signal from moving magnetizations in simple proportion to the velocity.

Early in the course of this project it was decided to use this last approach and pursue the method of phase encoding the signal of the flowing material. Phase encoding offers the advantage of relating a single property of the NMR signal to the flow rate. Factors, such as longitudinal and transverse relaxation, that affect the amplitude of the signal do not affect its phase. Since the phase of the NMR signal can be calculated using the classical model of the NMR phenomenon, it was felt that an understanding of the theory for flow measurements using phase encoding and a practical implementation would be straightforward. At the time, the dependence of signal phase on physical parameters of the spectrometer and magnetic system interactions was not fully understood or appreciated, so it is perhaps better at this point to provide some information on factors affecting the phase of the NMR signal before developing the theory of flow rate measurement by phase encoding.

3.1 The Phase of the NMR Signal

Changes in the phase of the signal from any elemental magnetization can be due to intentional manipulations of the applied magnetic field, such as the application of gradients in an imaging experiment, or can be the result of non-intentional interactions in the magnetic system, or can be the result of physical limitations in the NMR instrument itself.

An initial theoretical examination of the effect of non-uniform penetration of the RF magnetic field into the sample was made by Bottomley and Andrew (1978). This work describes how the RF magnetic field amplitude, its phase shift and power deposition vary with penetration depth in tissue models and suggests that, at high resonant frequencies (> 30MHz), changes in the amplitude and phase of the RF magnetic field in the sample would be significant enough to cause distortions of the image. However, later work by Bottomley et al. (1981, 1985) and Glover et al. (1985) demonstrates that these effects are not as severe as were first thought. Interestingly enough, work in this area since Bottomley and Andrew's paper has concentrated on only power deposition in tissue. To date the question of RF magnetic field phase changes within a sample, and the subsequent effect on the NMR signal recorded, is unresolved.

Norris (1985) showed that unwanted magnetic field gradients, which are inseparable from the imaging gradients, cause errors in phase images. These gradients produce magnetic field components in the direction of $\mathbf{B}_0$ that vary quadratically in the direction of their associated linear imaging magnetic field gradient. These unwanted gradients are a direct consequence of Maxwell's equations. In the case of the 2DFT spin echo imaging sequence, the amplitudes of the frequency encoding gradient pulses are not varied from one scan to the next. Hence the concomitant magnetic field gradient and the associated phase error is constant for each scan. However, the error due to the magnetic field gradient concomitant with the phase encoding gradient varies as the strength of the phase encoding gradient between scans. This leads to an appreciable change in the phase across a Fourier transformed scan line from one scan to the next. The effect will depend on the particular gradient pulse sequence and has not been investigated on the University of Alberta medium bore NMR unit.

The switching of magnetic field gradients induces eddy currents in the cryostat whose magnetic fields die away exponentially (Bendall, 1986). These additional magnetic fields can last tens of milliseconds. For example, an eddy current producing a magnetic field gradient in the z direction of 50×10^{-9} Tesla cm^{-1} of 20 milliseconds duration would produce a phase change (or phase gradient) of 0.267 radians cm^{-1}. These eddy currents can be compensated for electronically and through post-signal acquisition software processing, but it is very difficult to eliminate them completely.

Imperfections or physical limitations of the coils used to generate the RF and gradient magnetic fields can directly or indirectly influence the phase of the magnetization represented in images. For example, a gradient which varies linearly along any given line parallel to a cartesian axis can also vary radially within planes normal to that axis. This is illustrated in Figure 3.1. While this particular problem does not directly result in a phase error, it does restrict the method of gradient spin rephasing employed after a selective excitation sequence. This in turn affects the phase of magnetization moving along the slice gradient direction (refer to the section in Chapter 6 concerning slice selection).

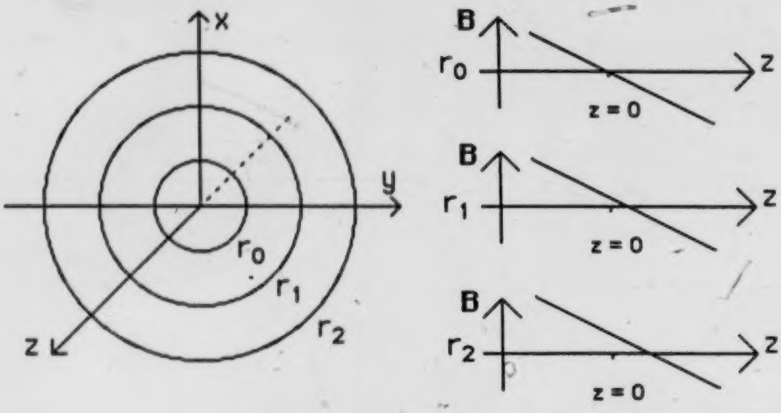


Figure 3.1 The radial dependence of a linear magnetic field gradient along the z axis. For each radius, the strength of the gradient magnetic field along z is plotted on the right.

These unwanted magnetic field interactions, inhomogeneities and distortions produce variations in phase images derived from two dimensional complex data sets. This complicates the analysis of phase dependent parameters in NMR experiments.

The pulse program sequence and its method of implementation plays a very important role in determining the phase of the NMR signal. As mentioned, phase variation across the image can be generated by the changes in the phase encoding gradient strength from one scan to the next. Such phase errors are manifest in the phase encoding direction of the image. A source of phase error in the frequency encoding direction of the image is in the sampling of the NMR signal. Consider a spin echo imaging sequence similar to that in Figure 2.19 with the time origin at the centre of the sampling window. Let the NMR signal, the spin echo, be given by a function g(t). The spin dephasing gradient pulse along the x axis between the 90 degree and 180 degree RF pulses can be varied in strength or duration to shift the position of the spin echo in the sampling window. A perfectly centred spin echo can be viewed as an even function in time. The complex Fourier transform of an even function results in a non-zero real frequency spectrum and a zero imaginary frequency spectrum. The phase spectrum is zero, hence, there is no phase shift. Fourier transform theory relates the position of g(t) in time with the phase of the frequency spectrum through the " time-delay " theorem. This theorem for a continuous Fourier transform is given by the Fourier transform operation pair

$$g(t-a) \Leftrightarrow e^{-i2\pi fa} G(f) \qquad 3.1)$$

where G(f) is the Fourier transform of g(t). The term $e^{-2\pi fa}$ is simply the phase for a given spectral component. Usually the spin echo is discretely sampled in a set time interval about the time origin (as described in section 2.10). The equivalent discrete Fourier transform operation pair is

$$g(kT-mT) \Leftrightarrow G\left(\frac{n}{NT}\right) e^{-i2\pi n\frac{m}{N}}.$$

3.2)

where T is the sample period

k is the sample number

N is the total number of samples

m is the number of time shifted sample points

It is clear that the term $e^{-i2\pi nm/N}$ will give phase shifts in m multiples of 2π across the discrete frequency spectrum. In the final image this will show up as linear phase shifts of 2π in the frequency encoding direction that are periodic with m.

Now, if the dephasing gradient pulse used to centre the echo were continuously adjustable, the continuous spin echo signal could be perfectly centred and this phase error eliminated. However, the gradient units in today's NMR imaging systems are controlled and set digitally. Thus, the strength of applied gradients is varied discretely. Using this system it is only possible to consistantly eliminate the discrete time shift for the spin echo. There can still be a small, continuous time shift of the signal. This leads to overall phase variations of less than 2π across the frequency spectrum. These shifts will vary from sample to sample due to differences in the sample volume and magnetic susceptibility. This has importance in the later discussion (Chapter 6) concerning the analysis of the phase images showing flow.

The use of linear magnetic field gradients to phase encode the NMR signal uniquely has been demostrated in section 2.10. The phase of the signal in a 2DFT imaging experiment is given by equation 2.56). In a more general form, the phase of the signal (with respect to the frequency of detection; see section 2.4) from the magnetization in an isochromatic plane normal to the gradient direction is

$$\theta_{\underline{r}} = \int_{\tau_1}^{\tau_2} \gamma \underline{G}_r(t) \cdot \underline{r}(t)\, dt \qquad 3.3)$$

where $\underline{G}_r(t)$ is the time dependent gradient function (Figure 3.2) and $\underline{r}(t)$ gives the time dependent location of the magnetization along $\underline{r}$. Hereafter the phase of the signal with respect to the frequency of detection will be referred to as simply the phase of the signal. τ_1 and τ_2 are points in time with respect to the time origin whose difference is the duration of the pulse.

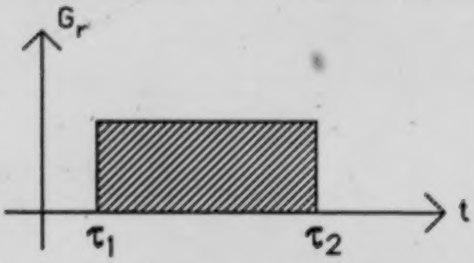


Figure 3.2 An ideal magnetic field gradient pulse.

For the case of a stationary magnetization at $\underline{r}$ and a rectangular gradient pulse, this equation reduces to

$$\phi_r = \gamma \underline{G}_r \cdot \underline{r} (\tau_2 - \tau_1) \qquad 3.4)$$

This is a very optimistic appraisal of the effect of a pulsed gradient. In reality, perfectly rectangular pulses cannot be generated. There is always a finite rise and fall time for the edges of the gradient pulse and a settling time before the gradient is static in time. The shape and duration of these edges and the settling times are governed by the method of gradient pulse generation (i.e., by the transfer function of the particular electronic circuits used). Thus, the exact shape of the gradient pulse is difficult to determine. The rise, fall and settling times place a lower limit on the minimum gradient duration. These factors restrict equation 3.4) to being only an approximation.

All the factors mentioned in the previous paragraphs contribute to the phase of the recorded NMR signal and of the eventual two dimensional complex image. Phase errors caused by such things as switched-gradient induced eddy currents and gradient asymmetry, have not been investigated in any detail in the work described this thesis. Much more theoretical work and experimental investigation is required to fully understand the cause of these errors. For instance, gradient generated eddy currents and imperfectly generated rectangular gradient pulses have been cited as sources for phase errors, yet no experimental evidence exists to distinguish the phase error generated by one source from that generated by the other.

For the work described in what follows, an empirical approach to the problem of phase errors was taken. Pulse programs that produced minimal phase errors in a phase image were developed through trial and error. Some effort was made to understand the problems associated with sequences that on paper worked, but that could not be effectively implemented. The final flow measurement pulse sequence implemented *evolved* from these investigations and the flow encoding theory to follow.

3.2 The Effect of Motion on a Two Dimensional Fourier Transform Image

Equation 3.3) gives the rotating frame relationship between the phase of an elemental magnetization in a magnetic field gradient and the position of that magnetization. If the position of the magnetization is time dependent during the gradient pulse, the phase acquired will be different from that which would have been acquired had the magnetization remained spatially static. This is the principle upon which van Dijk (1984), Bryant et al. (1984) and O'Donnell (1985), drawing on the work of Hahn (1960), have based their flow measurement imaging techniques. Their experimental work, and the work in this thesis, is restricted primarily to flow along the axis normal to the slice.

This section begins with a mathematical analysis of motion along all three cartesian axis based on the work of Deimling et al. (1986). Later, only motion normal to the slice will be analyzed. Based on this analysis and the aforementioned authors' work in flow measurements, a flow imaging sequence is defined in Chapter 6.

In the following analysis off resonance effects are ignored as well as longitudinal relaxation. It is assumed that the stationary magnetization is refocussed perfectly by the second G_z gradient pulse (Figure 3.3). The equations for the motion in the sample are

$$x(t) = x + v_x t \qquad 3.5)$$

$$y(t) = y + v_y t \qquad 3.6)$$

$$z(t) = z + v_z t \qquad 3.7)$$

That is, some elemental magnetizations in the sample have velocity components along one or more of the laboratory frame axis. The signal from the slice in the spatial frequency domain can be written as

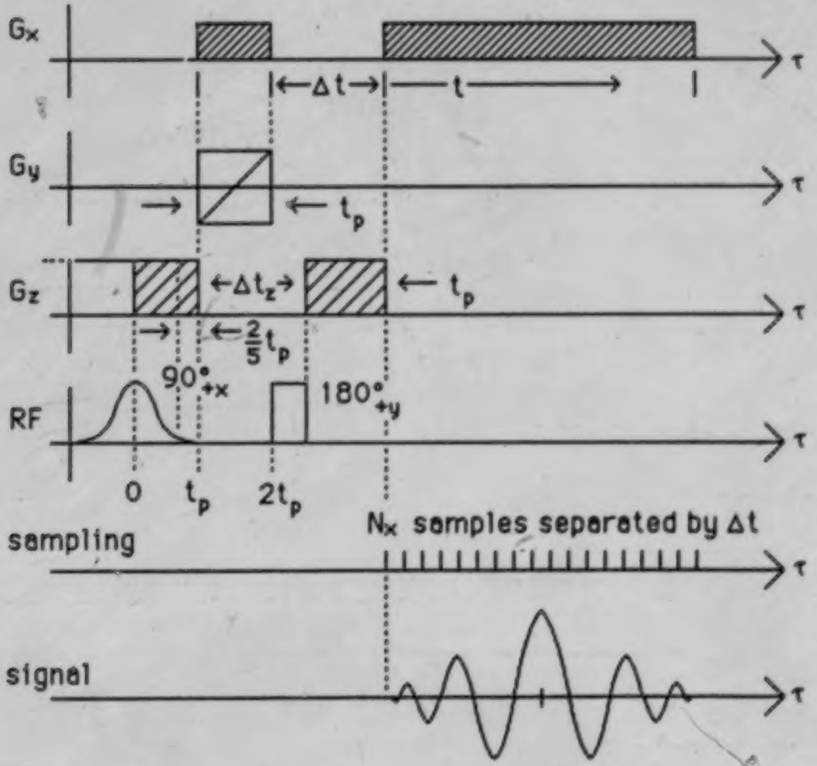
$$s(k_x,k_y) = \iint \rho(x,y) \exp(i\phi)\, dx\, dy \qquad 3.8)$$

where $\rho(x,y)$ is the spin density at (x,y) in the slice

ϕ is the phase of the signal from magnetization at (x,y)

k_x is $\gamma G_x(t-t_p)$, the spatial frequency coordinate in the x direction

k_y is $-\gamma G_y t_p$, the spatial frequency coordinate in the y direction



Gx
Δt
t
τ
Gy
t_p
τ
Gz
Δt_z
t_p
$\frac{2}{5}t_p$
τ
RF
90°_{+x}
180°_{+y}
τ
0
t_p
$2t_p$
Nx samples separated by Δt
sampling
τ
signal
τ

Figure 3.3 A typical spin echo pulse program.

k_x and k_y are, after some algebraic manipulations, the same terms as seen in equation 2.56). Deimling et al. (1986) consider only the terms involving motion along the x and y axes of the laboratory frame. The change in the phase term, ϕ, in equation 3.8) due to motion along the z axis is not easily quantified analytically. The problem lies with motion along the z axis during the time which the slice selective, shaped, 90 degree RF pulse and the slice gradient along the z axis are applied. Moving spins experience a change in the RF and gradient magnetic field amplitudes in time. This effect was simulated on a computer (see section 4.6.3). It was found that the change in the phase of the moving spins was equivalent to the phase change that stationary spins would experience during a gradient pulse of the same amplitude and in the same direction as the slice selection gradient, but only one fifth as long as the 90 degree RF pulse (or two fifths as long as the spin rephasing gradient pulse). Thus, using equations 3.3), 3.5), 3.6), 3.7), and the factor of two fifths mentioned above, ϕ is

$$\begin{aligned}\phi(t) = & -\gamma\int_{t_p}^{2t_p}(G_x x + G_x v_x \tau + G_y y + G_y v_y \tau)\, d\tau \\ & - \gamma\int_{0}^{t_p} G_z z\, d\tau - \gamma\int_{\frac{3}{5}t_p}^{t_p}(G_z v_z \tau)\, d\tau \\ & + \gamma\int_{2t_p+\Delta t}^{2t_p+\Delta t+t} G_x (x + v_x \tau)\, d\tau + \gamma\int_{t_p+\Delta t_z}^{t_p+\Delta t_z+t_p} G_z (z + v_z \tau)\, d\tau\end{aligned} \qquad 3.9)$$

Note that the time origin has been shifted in this analysis from that which was used in section 2.10. Since only the phase is considered here, this time shift is not important. Equation 3.9) reduces to (see Appendix A)

$$\phi = xk_x + yk_y + v_x k_x (3t_p + \Delta t) + v_x \gamma G_x (t_p^2 + \Delta t\, t_p) + v_y k_y \frac{3}{2} t_p + \frac{v_x k_x^2}{2\gamma G_x} + \gamma G_z \frac{v_z}{2} (\frac{59}{25} t_p^2 + 2t_p \Delta t_z) \qquad \text{3.10)}$$

Using equation 3.10) as the phase term in equation 3.8) and taking the two dimensional Fourier transform yields terms similar to those of equation 2.67) (that is, a mapping of the elemental magnetizations in two dimensions) plus additional terms due to motion. The velocity terms linear with k_x and k_y cause a linear shift of the image data points that correspond to points in the object with velocity components in the x or y directions respectively. Considering x and y velocity components that are approximately the same in magnitude, the $v_x k_x(3t_p + \Delta t)$ term, that is, motion along the readout or frequency encoding direction, will dominate this effect since it has a large time multiplicand. For example, for a velocity of $v_x = 50$ cm/s and typical pulsed magnetic field gradient amplitudes and durations, the shift in an image derived from the first echo would be about 9 mm. The velocity term quadratic with k_x will cause a slight blurring along the x direction due to convolution of the data points with a sinc-like function. The two terms in equation 3.10) that are functions of neither k_x nor k_y modify the phase of the data points where there is velocity component along either the x or z directions in the object.

Van Dijk (1984), Bryant et al. (1984) and O'Donnell (1985) have based their pulse sequences for flow quantification on such phase effects. The objective of their method was to uniquely differentiate moving or flowing tissue from non-moving tissue. The work in this thesis will concentrate on motion due to blood flow in large vessels perpendicular to the imaging plane, i.e., along the z direction.

All three authors begin with the naïve assumption that immediately after the slice selection sequence of a spin echo sequence, the phase of all moving and stationary magnetizations is zero. This is not completely true as the previous development has shown (see equation 3.10), but it does help illustrate the following argument. This assumption, plus the fact that only motion along z is considered, reduces equation 3.10) to

$$\phi = xk_x + yk_y \qquad 3.11)$$

This equation has no dependence on motion along the z direction. To induce a change in the phase due to motion another gradient pulse applied along z can be added to the sequence. If the position of the magnetization is given by equation 3.7), then the phase contibution due to motion would be

$$\begin{aligned}\phi_z &= \gamma\int_0^\tau G_z\,(z + v_z t)\, dt \\ &= \gamma G_z(z\tau + \frac{v_z}{2}\tau^2)\end{aligned} \qquad 3.12)$$

Notice that the stationary magnetization also acquires a phase offset. To remove this, a term $-z\tau$ is needed. This is added by applying a second consecutive gradient pulse along z such that the area of the two pulses over time is zero (see Figure 3.4).

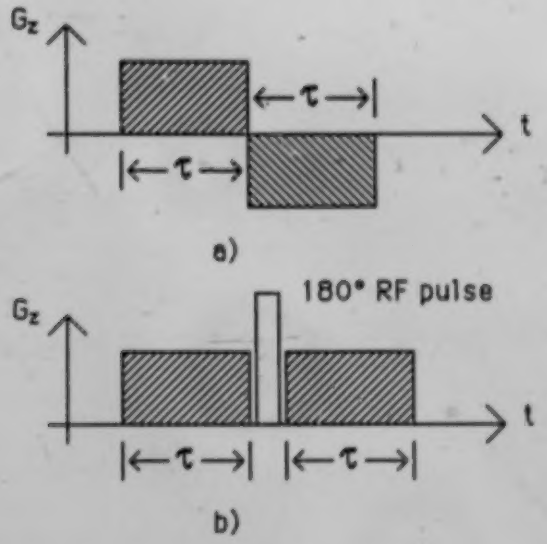

G_z
$\leftarrow \tau \rightarrow$
t
$\leftarrow \tau \rightarrow$
a)
180° RF pulse
G_z
t
$\leftarrow \tau \rightarrow$
$\leftarrow \tau \rightarrow$
b)

Figure 3.4 Balanced gradient pulses.

The gradient pulse set in Figure 3.4 b) can be effectively considered to be continuous if it is assumed that $\tau >> t_{180°} = \pi/(\gamma B_1)$. Such gradient pulse sets are referred to as " balanced gradients ". Equation 3.12) now becomes

$$\begin{aligned}\phi_z &= \gamma\int_0^{\tau} G_z\,(z + v_z t)\,dt - \gamma\int_{\tau}^{2\tau} G_z\,(z + v_z t)\,dt \\ &= \gamma G_z\,(z\tau + \frac{v_z}{2}\tau^2 - z\tau - \frac{3v_z}{2}\tau^2) \\ &= -\gamma G_z v_z \tau^2\end{aligned}$$

3.13)

Notice that the term for the non-moving magnetization is gone and that only a phase term proportional to velocity remains. Thus, any region of the object in the imaging plane with constant velocity components along the z axis will produce a change in the phase of the image pixels at those points. This change in phase can be measured and correlated to the motional velocity using equation 3.13). The sign of the phase change will depend on the direction of the motion along z and the temporal order of the motion encoding pulses (i.e., whether the positive lobe of the balanced gradient pulse set occurs before the negative lobe or after. See Figure 3.4). At the time of image reconstruction a phase image is generated by taking the arctangent of the imaginary component divided by the real component of each complex point in the image (θ = arctan[imaginary/real]).

A problem that is common to any cyclic function is that of aliasing. In this case, phase changes that are multiples of π are indistinguishable from those within a $\pm\pi$ range about zero. This restricts the maximum velocity that can be measured using a single balanced gradient pulse. The balanced gradient pulse must be defined so that the phase encoded for the maximum velocity component likely to be present is between $\pm\pi$. This leads to a problem with very low velocities. The phase change at these rates may be indiscernible from fluctuations in phase due to noise and the aforementioned phase error sources.

The earlier assumption that the phase of the signal is zero immediately after the slice definition sequence is of course invalid. To the contrary, for flow along the slice gradient direction, the phase acquired will be quite large because of the duration and magnitude of typical slice definition sequence gradients. At the University of Alberta medium bore

NMR unit, the slice defining sequence used resembles the sequence shown in Figure 3.3. This sequence produces, for even low flow rates, a substantial phase offset at the onset of the spin echo. The term responsible for this, given in equation 3.10), is $v_z\gamma G_z(\{59/50\}t_p^2 + t_p\Delta t_z)$. However, placing a balanced gradient pulse between the 90 degree pulse and the 180 degree pulse in the sequence can reduce or eliminate this offset (see Chapter 6). Setting the right hand side of equation 3.13) equal to the above term gives an estimate of the magnitude of the balanced gradient pulse needed. That is,

$$G_z\left(\frac{59}{50}t_p^2 + t_p\Delta t_z\right) = G_{z_{compensation}}\tau^2 \qquad 3.14)$$

This is a rough estimate since the shape of the gradient pulses is in reality not rectangular. However, experimental results (see Chapter 6) have shown that equation 3.14) is a good first order approximation.

3.3 The Effect of Pulsatile Flow

Up to this point it has been assumed that the velocity of the movement or flow has been constant. Physiologically, this is rarely true. Pulsatile motion of the blood in arteries can generate velocities from 0 cm/s to peaks of 150 cm/s (in the centre of the ascending aorta during systole) (McDonald, 1974). The upper thorax is the region of highest blood flow velocities. The measurement of arterial blood flow using NMR imaging, or for that matter, any upper thorax NMR imaging application, is possible only using cardiac gating. Even with gating there would be acceleration components of flow present. Gating would ensure that the motion was at least the same for each scan of the imaging sequence. Cardiac gating was not available at the time of this project, so constant flow, as would be found in some veins, was concentrated on.

The effect of pulsatile flow on an image has been mathematically investigated in Appendix B. In short, it can be said that constant velocity motion along the axis normal to the imaging plane affects only the phase of the pixels in the image representing the region of motion in the object. If, however, the velocity fluctuates in time due to pulsatile motion or flow turbulence, artifacts are generated in the phase and magnitude images in a band in the phase encoding direction which is the width of the region in the object where the fluctuating motion is present.

This analysis has implications in the quantification of NMR parameters in images. Such artifacts, which affect the modulus or magnetization density image, can lead to incorrect measurement of such quantities as T_1 and T_2. In this thesis, artifacts in phase images due to fluctuating velocity help to indicate a non-constant velocity condition. This is important since the work in this thesis has been based on the assumption of constant flow velocity.

4. Computer Simulation of NMR Experiments

4.1 Background

Computer simulations in various forms have been finding widespread application in the modelling of NMR phenomena and experiments. For applications of NMR in medicine and biochemistry, these simulations are invariably based on the phenomenological Bloch equations. The flexibility of numerical solutions of these equations via computer simulation has been demonstrated by several authors (Locher, 1980; Constantinesco et al., 1984; Wood and Henkelman, 1985; Brandes and Kearns, 1986; Summers et al., 1986). Applications have ranged from selective irradiation simulations (Locher, 1980; Brandes and Kearns, 1986) to full scale imaging experiments (Summers et al., 1986). The computer simulation discussed in this chapter fits somewhere in the middle.

The initial impetus behind this simulation was a desire to understand basic NMR phenomena, such as the interaction of the magnetization with RF magnetic fields and magnetic field gradients. This led, eventually, to experiments involving the effects of motion during a NMR experiment on the acquired signal phase.

From the outset the simulation was written to follow as closely as possible the sequence of events in a typical NMR experiment. These events or steps are, in broad terms,

1) The definition of the pulse sequence, the sequence of scans and the object.
2) The start of a scan.

3) The acquisition of the NMR signal and the temporary storage of the signal (if desired, permanent storage of the signal).
4) The end of the scan.
5) If necessary, return to step 2).
6) The processing of the NMR signal data set. Usually this involves a one or two dimensional Fourier transformation of the data set.
7) End of the experiment.
8) Post-processing of the data.

This simulation was left purposefully open ended. By this it is meant that the program is easily reconfigured to run different experiments. New modules simulating parameters not yet included in the package are easily added. With moderate changes to the modules, the dimension of the object may be changed.

4.2 Simulation Methodology

The simulation package consists of modules that control either program flow or NMR phenomena modelling. These modules can be arranged to mimic a pulse program and are invoked from a main control program. This allows the module library to stand apart from the main program. Only the modules required for a particular experiment are linked with the main program, thus saving program code and run time code space. All interations are modelled in the rotating reference frame.

The object is modelled as point source magnetizations in a rectangular three dimensional cartesian coordinate system. For each point values of T_1 and T_2 are specified. In the current system, the magnetic fields $\underline{B}_0$ and $\underline{B}_1$ are perfectly homogeneous and static over the time which they are applied. The object space can be reduced in dimensionality or size through moderate program modification. Thus a two or one dimensional object is possible. This is desirable in that it reduces the total running time of the experiment.

The duration of some events, e.g. RF pulses, is specified at the time of the module call and controlled by the module. Other event durations, such as gradient pulse durations, are controlled by the main program. Multi-scan experiments are accomplished by having the main program loop through the module sequence.

Many events can be modelled using an iterative solution of the Bloch equations (see Appendix C) or by an explicit solution. For example, transverse relaxation is modelled by an iterative solution of the Bloch equations which includes a transverse relaxation term or by using the relation $M(t) = M_0 e^{-t/T2}$, where M(t) is the transverse magnetization magnitude in time and M_0 is the transverse magnetization at t=0. Whatever the method of solution, program control is based on " slices " of time or discrete time steps. The size of these steps depends on the method of solution for a given parameter model.

All data acquisition is done by ideal sampling. Sampling can involve saving the entire state of the object space or saving a vector sum of the magnetization components (M_x, M_y, M_z) over the entire object space. During the sampling operation the simulation is static, i.e., sampling is done instantaneously. Saving the data in this way is analogous to the Quadrature Phase Sensitive Detection (QPSD) method mentioned in section 2.4. If a FID or spin echo is to be recorded, an acquisition module is invoked which records the vector sum of all the magnetizations at regular intervals for a set number of samples.

Data processing during the simulation is limited to fast Fourier transformation of an acquired signal. The result of this operation can be saved by the main program for later analysis outside the simulation.

After the " scan " sequence of modules has been executed by the main program, the simulation is terminated by halting the program.

4.3 Module Functional Descriptions

A description of the pupose and action of the modules is given. The modules are grouped according to function. The module listings appear in Appendix D.

4.3.1 Pulse Program and Bloch Equations Modules

DM_DT_PLUSX	This module solves the Bloch equations in the rotating frame of reference using a fourth order Runge Kutta iterative method. The RF magnetic field is considered to be along the positive x axis of the rotating frame. The location of the magnetization in space and the time interval over which to solve the equations are passed to the module. The form of the Bloch equations and the Runge Kutta method of solution are given in Appendix C.

DM_DT_MINUSX	The same as above except that the RF magnetic field is oriented along the negative x axis of the rotating frame.
DM_DT_PLUSY	The same as above except that the RF magnetic field is oriented along the positive y axis of the rotating frame.
DM_DT_MINUSY	The same as above except that the RF magnetic field is oriented along the negative y axis of the rotating frame.
INIT_MAGNETIZATION	This module specifies the object magnetization, T_1 and T_2 value distributions. It is usually invoked at the beginning of the experiment.
CLEAR_GRADIENTS	This module zeros the gradient values over the entire object space.
GRADIENT	This module sets a specific gradient using a passed gradient direction (x, y or z) and passed maximum and minimum values of the gradient at either end of the object along that direction. The gradient is set to vary linearly over the sample. New values are added to previous gradient values at each point in the object space. Therefore, it is important to consider previous invocations of this module. Old gradient values may still remain. This is much like the method of gradient specification on the spectrometer. The gradient values for all three directions must be specified at every gradient switching point. Coupled with the

	CLEAR_GRADIENTS module, these modules effectively control gradient pulses in the simulation.
NO_RF_NO_GRADIENTS	This module models the magnetic system when no RF or gradient magnetic fields are present. Only relaxation processes are considered. Each magnetization in the object is adjusted by calculating the assumed exponential decay for each magnetization component over the duration passed to the module.
NO_RF_GRADIENTS_ON	This module models the change in the magnetic system under the influence of a linear magnetic field gradient. The time interval passed to the module is broken up into discrete steps. For each step, the DM_DT_PLUSX module is invoked for each point in the object space to solve the change in the magnetization. The orientation of the RF magnetic field matters not as it is set to zero. Since the iterative solution of the Bloch equations is used, relaxation effects are implicitly accounted for.
NO_RF_GRADIENTS_ON_IDEAL	This module models the change in the magnetic system under the influence of a linear magnetic field gradient. The total precession of the magnetizations over the time interval passed to the module is calculated trigonometrically. Each magnetization is adjusted

	accordingly. In this case relaxation is calculated explicitly.
HARD_PULSE	A rectangularly modulated RF pulse is modelled using the iterative solution of the Bloch equations. For the time duration and the tip angle passed, the amplitude of the RF magnetic field is calculated and then set. The time duration is broken into discrete steps if necessary and, based on the direction passed, the appropriate DM_DT (±x, ±y) module is invoked at each step for each magnetization in the object space. At the end of the module execution the RF field is set to zero.
HARD_90_X_IDEAL	The magnetization components in the y-z plane of the rotating frame are explicitly rotated 90 degrees counter-clockwise when looking along the x axis. The rotation is done trigonometrically. Again, relaxation is adjusted explicitly.
HARD_180_Y_IDEAL	The magnetization components in the x-z plane of the rotating frame are explicitly rotated 180 degrees counter-clockwise when looking along the y axis. The rotation is done trigonometrically. Again, relaxation is adjusted explicitly.
SOFT_PULSE	A shaped (i.e., other than a rectangular envelope) RF magnetic field pulse is modelled by this module. The orientation of the RF magnetic field, the duration of the pulse, a vector

	of points specifying the shape and magnitude of the pulse and the size of the vector are passed to the module. The time duration is broken into as many intervals as there are shaped pulse points in the vector. The RF magnetic field amplitude is set to the value specified in the vector for the duration of the time interval corresponding to that vector point. Then, if necessary, the time interval is divided into subintervals short enough to accurately model the change in the magnetizations using the appropriate DM_DT module as indicated by the passed field direction (±x, ±y). After the shaped pulse has been applied, the RF magnetic field amplitude is set to zero.
GAUSS	This module calculates the amplitudes specified in the shaped pulse vector for a 90 degree gaussian shaped RF pulse. The number of points to be used for the pulse and the duration of the pulse are passed. The amplitudes are set by iteratively solving for the condition specified by Sutherland and Hutchinson (1975). They state that the effective area under the gaussian pulse shape envelope must equal $\pi/(2\gamma)$.
GSINC	The same as above except that the pulse shape is a gaussian windowed sinc pulse as specified by Locher (1980).

HSINC	The same as above except that the pulse shape is a Hanning windowed sinc pulse.
GAUSS180 GSINC180 HSINC180	The same as the correspondingly labeled modules above except that pulse area condition is π/γ. Hence these are 180 degree pulses.

4.3.2 Signal Acquisition

ACQUIRE_SIGNAL	This module is used to record a sampled waveform over a set duration for a specified number of points. The time duration is divided into intervals. Over each interval the module NO_RF_GRADIENTS_ON is invoked. Hence the signal can be acquired under the influence of a gradient. At the end of each interval separate sums of the x and y magnetization components over the entire object space are stored as a complex number ($\Sigma M_x + j\Sigma M_y$). This number is pushed onto a stack which is passed back to the main program for later processing.
ACQUIRE_SIGNAL_IDEAL	The same as above except that NO_RF_GRADIENTS_ON_IDEAL is invoked instead.

4.3.3 Data Manipulations

INIT_STACK	This module initializes the stack before the invocation of an acquire module.
PUSH	This module pushes complex data onto the top of the stack.
PULL	This module removes data from the top of the stack.
FFT1,FFT2	This module performs a one dimensional complex Fourier transform (FT) operation. The complex data must be passed in two vectors, one containing the real part of the complex numbers and the other containing the imaginary part. Invocation of FFT1 prepares the routine for the calculation. Invocation of FFT2 performs the FT. The results are left in the original data vectors and the input data is destroyed. Note that it is up to the programmer to normalize the results by the number of passed points (1/N).

4.3.4 Program Constants

Program constants and data structures are specified at the beginning of each module and at the beginning of the main program. Parameters and structures that are likely to change from one experiment to the next are specified outside the modules and the main program. They are included through the use of the VAX-FORTRAN " INCLUDE " statement in the source code. This looks for the appropriate file of parameter and data structure definitions and places that code in-line with the source code at compile time. For example, the statement, INCLUDE "BLOCHD1D.FOR", would place the contents of BLOCHD1D.FOR inline with the source code at the point the INCLUDE statement appears. This scheme ensures that changes in the data structures and program parameters are automatically applied to all modules affected, thereby easing program modification and maintenance.

4.4 Simulation Constraints and Limitations

The limiting factor determining the usefulness of the simulation is the time it takes to run. As has probably already been realized, the so called " ideal " modules were included to reduce running time. The maximum time slice for an invocation of the DM_DT series of modules cannot really be lengthened without causing unacceptable errors. A factor closely related to the total run time is the size of the object. Even experiments involving small objects can take many hours to run. For example, all the experiments described in this chapter utilize a one dimensional sample. Due to the nature of the experiments, no additional information could be obtained for larger objects.

As mentioned earlier, the magnetic fields in the simulation are assumed to be homogeneous and static. This could be changed by including matrix elements corresponding to points in the object space that model inhomogeneity, field drift and other imperfections. These elements would be included in any calculations involving the magnetic field.

The Bloch equations used in this simulation model only the interaction of the magnetization with the magnetic fields $\underline{B}_0$ and $\underline{B}_1$ and the effects of longitudinal and transverse relaxation. Additional terms could be added to the Bloch equations to model such things as diffusion (Torrey, 1956). This, of course, would involve reworking the transformation presented in Appendix C.

4.5 Simulation Testing

In general, any numerical solution to a problem produces errors associated with the method of solution and the computational methods. Since this simulation has been implemented in FORTRAN on a digital computer (Digital Corp. VAX-11/750), round-off errors are present in many of the computations. Round-off error occurs because the representation of real numbers is only approximate due to the finite size of the computer word and the use of the binary number system. Thus, many calculations involve approximate representations of the actual numbers. The results of these calculations may be used repeatedly, thus propagating the error. However, due to the variability of the misrepresentation of real numbers and the unpredictability of the sequence of calculations from one experiment to the next, a quantitative measure of the overall error in the simulation due to round-off is not easy.

The main source of error in the simulation is in the modules which utilize the fourth order Runge Kutta method to solve the rotating frame Bloch equations. This solution, as specified in Appendix C, involves several iterations per time interval. The maximum simulated time interval allowed is 100 microseconds and the number of iterations is 128, therefore, in a typical NMR experiment of 100 milliseconds, there can be many intervals. It is easy to see that the bulk of the computations are made by these modules. Hence round-off errors will be most prevalent in these modules.

The error associated with the fourth order Runge Kutta method itself is of the order h^4 (Burden et al., 1981), where h is equal to the interval width divided by the number of iterations (see Appendix C). For example, if the interval were 100 microseconds and the number of iterations is 64, then h^4 is $\approx 6 \cdot 10^{-24}$ seconds.

Unfortunately, the quantitative contributions of the methodological and round-off errors to the total error in the simulation are not known. In any case, as mentioned, the variablility of the experiments simulated makes it difficult to judge the errors.

The above discussion had consequences in the initial testing of the simulation. To test the correctness and the accuracy of the four modules employing the Runge Kutta method, the effect of 90 degree RF magnetic field pulses on a small object in thermal equilibrium was simulated. Ten point source magnetizations along the z axis were initialized to have a magnetization value at t=0 of $\underline{M}(0) = \underline{k}M_0$. M_0 was arbitrarily chosen to be 100.0. After the application of a 90 degree RF pulse along, say, the x axis of the rotating frame, the magnetizations should be $\underline{M}(t_{90}) = \underline{j}M_0$. In the ideal situation there should be no magnetization components along either the x or z axis. The two variables available in the simulation in this test were the time interval size, which is also the duration of the 90 degree RF pulse, and the number of iterations used by the Runge Kutta method. These were varied until an acceptable error in the magnetization components after the 90 degree pulse was arrived at (i.e., until an acceptable minimum M_z value was found). Some results of this test are tabulated in Table 4.1. This test resulted in an empirical choice of the maximum time interval and the number of iterations used by the Runge Kutta method (i.e., a maximum time interval or 100 µS and 128 iterations).

90° pulse duration	Number of Runge Kutta iterations for a minimum M_z	M_x	M_y	M_z
100mS	64	0.000	91.293	7.563
10mS	64	0.000	99.075	0.792
1mS	64	0.000	99.907	0.0795
100µS	128	0.000	99.991	0.00795
10µS	64	0.000	99.999	0.000794
1µS	128	0.000	99.999	0.0000776
100nS	256	0.000	99.999	0.00000596
10nS	64	0.000	100.00	-0.00000073
1nS	64	0.000	100.00	-0.00000144

Table 4.1 Some iteration and time slice efficiency comparisons.

After setting the number of iterations and the maximum time interval, it remained to test the gradient modules and off-resonance effects. Reproduction of the off-resonance effects described in the first part of Locher's paper (1980) was deemed to be a suitable test. Locher simulated the effect of a rectangularly modulated 90 degree RF pulse in the presence of a linear gradient. The simulated object was placed in a strong magnetic field $\underline{B}_0$. A RF magnetic field, $\underline{B}_1$, was applied for a duration τ along the x axis of the rotating frame. Throughout the application of the RF pulse a linear magnetic field gradient in the direction of $\underline{B}_0$ was present. By examining the effective magnetic field experienced by the magnetization in the rotating frame, Locher derived the following expressions for the magnetization components at the point in time at which the RF pulse ended. These were

$$M_y/M_0 = \sin(\theta g)/g \qquad 4.1)$$

$$M_x/M_0 = f[\,1 - \cos(\theta g)\,]/g^2 \qquad 4.2)$$

$$M_z/M_0 = [\,f^2 + \cos(\theta g)\,]/g^2 \qquad 4.3)$$

where

$$f = (\,B_0 - \omega/\gamma\,)/B_1$$

$$g = (\,1 + f^2\,)^{1/2}$$

θ is the flip angle at $f = 0$ and f is the spin resonance frequency in units of $B_1\gamma/(2\pi)$. M_x, M_y and M_z are the rotating frame components of the magnetization. The thermal equilibrium magnetization magnitude is M_0.

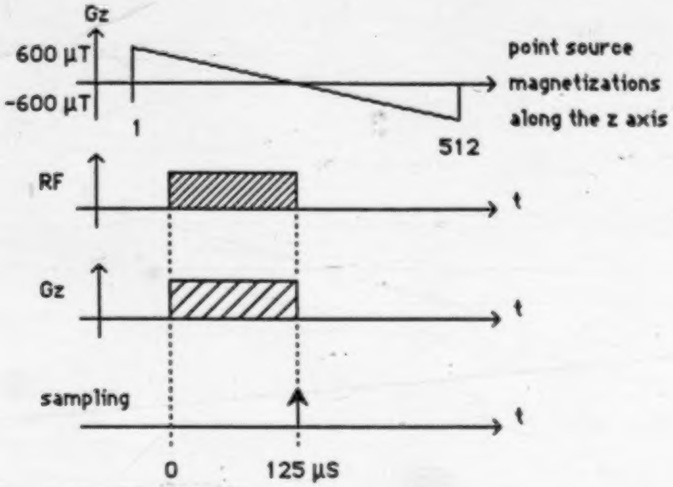


Figure 4.1 The pulse program used as a model for the simulation experiment.

The object used in the simulation consisted of 512 point source magnetizations along the z axis. A gradient of 2.34375 microTesla per point was established along the z axis. A rectangular 90 degree RF pulse of 125 microseconds duration was applied to this system and the magnetization components of each point were recorded as soon as it ended. The sequence is given in the pulse program in figure 4.1.

The results of the simulation are plotted in figures 4.2 through 4.6. Data points from the computer simulation were compared with the exact values given by the above equations (calculated by the computer program in Appendix E). For a randomly selected set of twelve point source magnetizations, the largest average absolute error in any of the magnetization components, M_x, M_y and M_z, was 0.386. The equilibrium magnetization value was 100.

These experiments were the extent of the simulation testing. Time did not allow for more extensive testing.

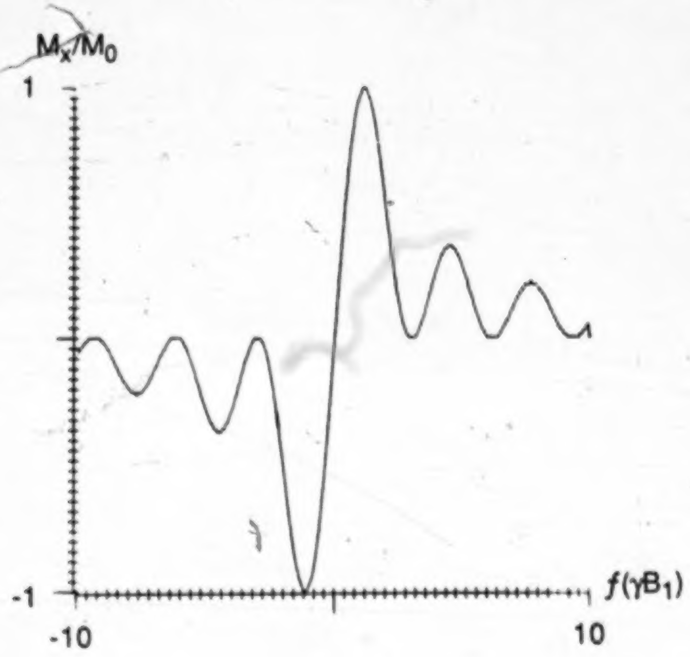


Figure 4.2 The dependence of M_x on spin resonance frequency.

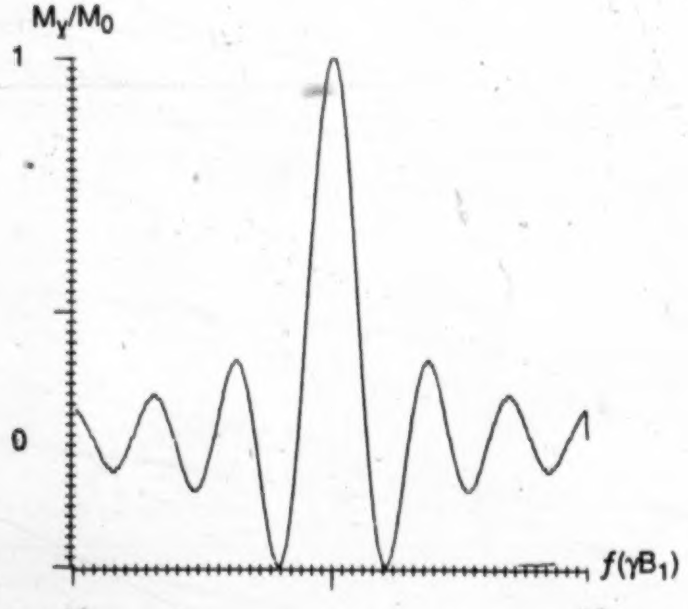


Figure 4.3 The dependence of M_y on spin resonance frequency.

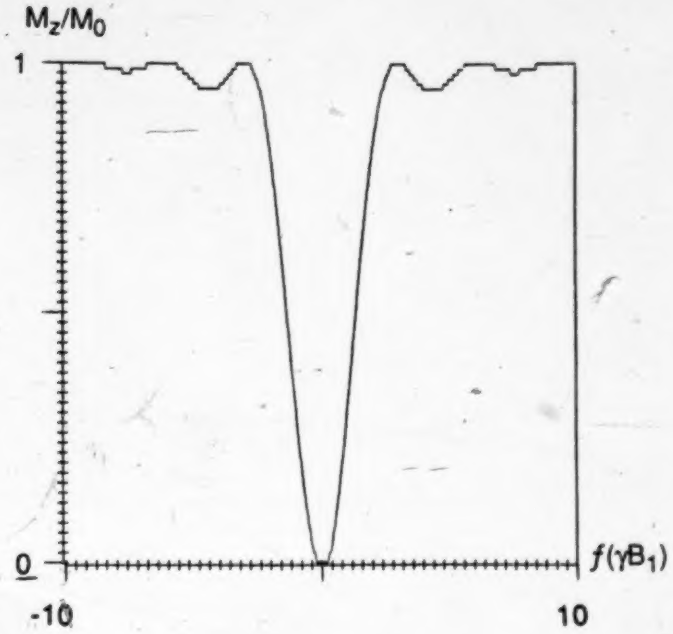


Figure 4.4 The dependence of M_z on spin resonance frequency.

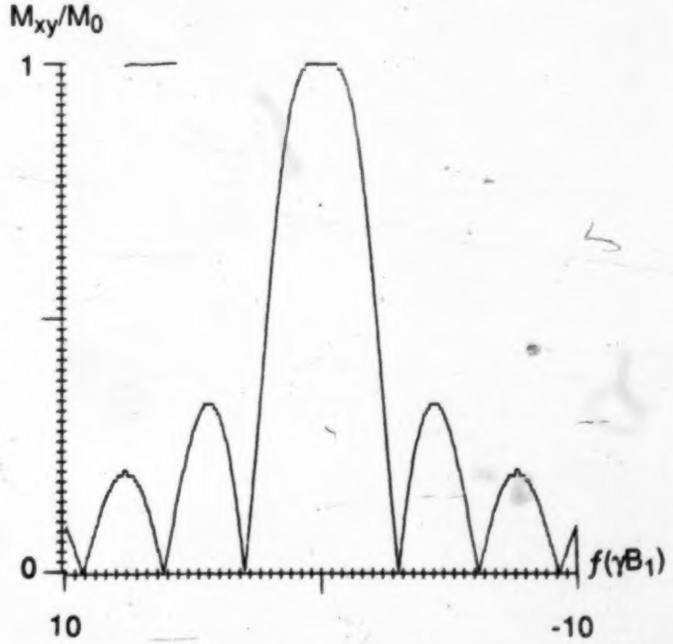


Figure 4.5 The dependence of M_{xy} on spin resonance frequency, where $M_{xy} = (M_x^2 + M_y^2)^{1/2}$.

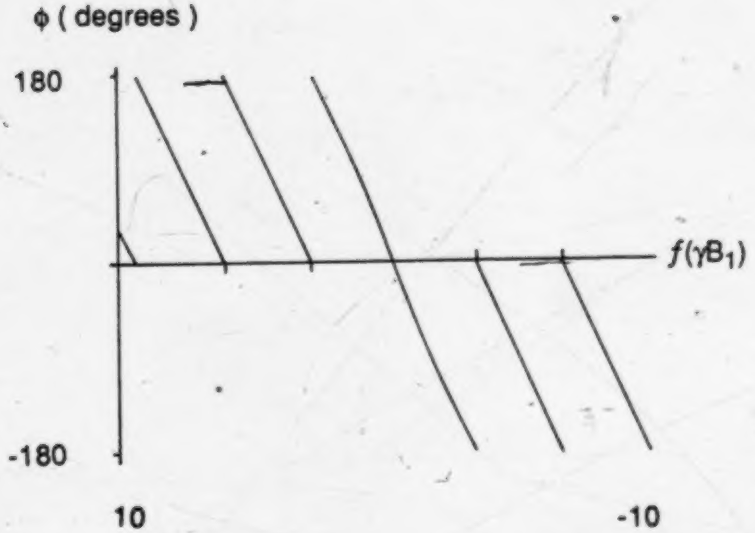


Figure 4.6 The dependence of the transverse magnetization phase on spin resonance frequency. The phase is given by

$$\tan\phi = -M_x/M_y.$$

4.6 Flow Experiment Simulations

The results of the computer simulation experiments presented in this section were used to investigate aspects of flow measurement theory as presented in Chapter 3. The first two experiments simulated the effect of balanced gradient pulses on the phase of the NMR signal from magnetization moving at a constant velocity. The third experiment simulated the effect on the phase of the spins of constant velocity motion along the direction of a gradient during a Gaussian shaped 90 degree RF magnetic field pulse.

The object in all three experiments was comprised of 512 point source magnetizations along the z axis. Motion was generated by shifting the modelled magnetizations by one position in the data structure at regular intervals in the experiment. Velocity was defined as the total number of data structure locations or points through which the modelled magnetizations were to be moved divided by the total time in the experiment over which motion was present. This gave units of points per second. The time between successive shifts of the points was given by the reciprocal of the velocity (i.e., seconds per point moved). To accomplish the shift of the magnetizations the modules used in the experiment had to be modified. Each module kept track of the time elapsed in the experiment since the onset of motion. When a multiple of the time between point shifts was reached, a separate module to shift the points was called.

4.6.1 Experiment I

The effect of balanced gradient pulses on the phase of the NMR signal arising from moving magnetization was simulated. The pulse sequence used as a model is shown in Figure 4.7. The rectangularly shaped 90 degree RF magnetic ~~field~~ pulse was used to uniformly excite the magnetization. That is, each point source magnetization was rotated from its thermal equilibrium orientation, along the z axis, through 90 degrees onto the y axis of the rotating frame.

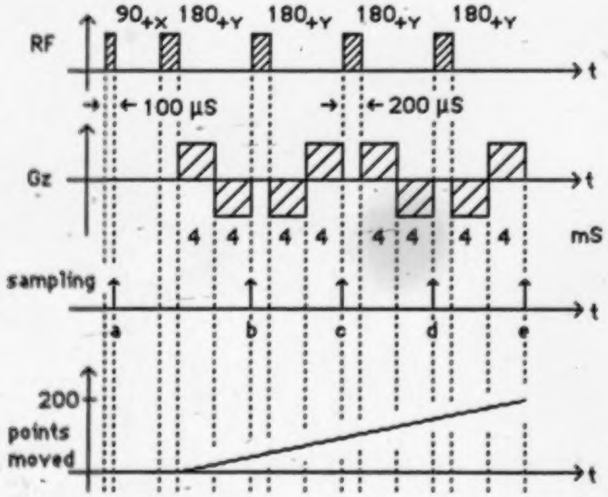


Figure 4.7 The pulse program modelled in experiment I.

Notice that the point source magnetizations were stationary during the 90 degree pulse. This excluded interactions between the moving magnetization and the 90 degree pulse allowing for clearer observation of the effect of the balanced gradient pulses. Each gradient pulse was of magnitude 761.953 · 10^{-12} Tesla/point. Data sampling (points a through e) was instantaneous, i.e., no experimental time passed while data points were being recorded. Each data point consisted of the x, y and z components of a particular point source magnetization. Data points were recorded for all the magnetizations. From this data the phase of the overall transverse magnetization with respect to the y axis of the rotating frame was calculated (i.e., $\phi = \tan^{-1}[M_x/M_y]$). The results for each sample point are given in Table 4.2.

Sample point	Measured Phase	Absolute Phase Change
a	0.00	--
b	1.17	1.17
c	-2.38	1.21
d	3.01	0.63
e	-4.30	1.29
Average phase change per sample =		1.07

Table 4.2 Results from experiment I.

It must be remembered that the 180 degree RF pulses act to invert the sign of the signal phase. Using the above simulation parameters and equation 3.13), the change in the phase of the NMR signal due to a balanced gradient pulse is

$$\begin{aligned} \phi &= \gamma G_z v_z \tau^2 \\ &= \frac{2\pi \cdot 100 \cdot 10^{6}}{2.35} \cdot \frac{180}{\pi} \cdot 761.953 \cdot 10^{-12} \cdot \frac{200}{32.6 \cdot 10^{-3}} \cdot 16 \cdot 10^{-6} \\ &= 1.145 \text{ degrees} \end{aligned}$$

The simulated phase changes compare favourably with this result. The differences between the two results can be possibly explained by the method of motion generation in the simulation (i.e., the interval between shifts was not exact. Also, as can be seen in Figure 4.2, the motion was not continuous, but stepped.). Round-off errors also played a part. For example, the values of the magnetization components were real in the simulation, but were rounded to the nearest integer when sampled and stored.

4.6.2 Experiment II

This experiment was the same as the first experiment except for the method of magnetization excitation. The effect of a more realistic " slice-definition " sequence on the phase of the NMR signal from moving magnetization was simulated.

The 90 degree RF magnetic field pulse in this experiment was Gaussian shaped and of 7.68 milliseconds duration. During the RF pulse there was a linear magnetic field gradient of $232.061 \cdot 10^{-9}$ T/point along the z axis. Immediately following this RF pulse the magnetic field gradient was reversed in direction and left on for an additional 3.84 milliseconds. At this point in time the magnetizations were spread over an angle of approximately 20 degrees about the y axis of the rotating frame, giving a net magnetization along this axis. Following this " slice-definition " sequence the pulse sequence is the same as that shown in Figure 4.7 following the rectangular 90 degree RF pulse. The pulse sequence modelled in this experiment is shown in Figure 4.8. Once again balanced gradient pulses along the z axis were used to alter the phase of the NMR signal between 180 degree RF pulses.

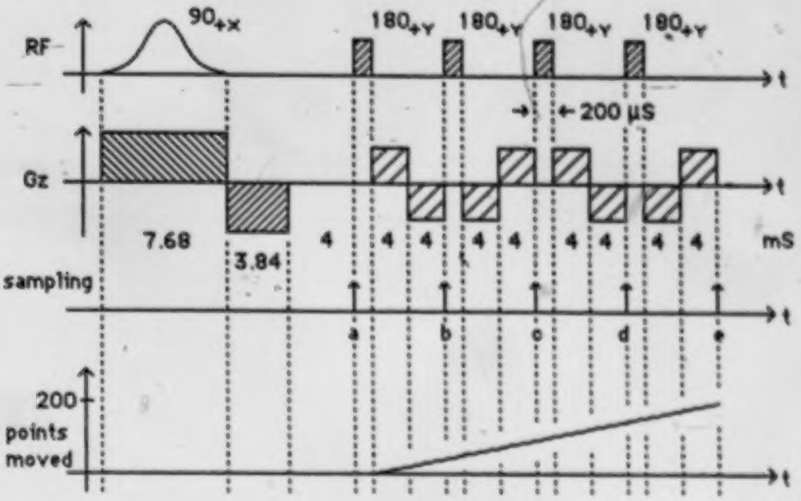

RF
90_{+x}
180_{+Y}
180_{+Y}
180_{+Y}
180_{+Y}
t
200 µS
Gz
t
7.68
3.84
4
4
4
4
4
4
4
4
4
mS
sampling
a
b
c
d
e
t
200
points moved
t

Figure 4.8 The pulse program modelled in experiment II.

Sample point	Measured Phase	Absolute Phase Change
a	7.52	--
b	-6.36	1.16
c	5.24	1.12
d	-4.05	1.19
e	2.99	1.06
Average phase change per sample =		1.13

Table 4.3 Results from experiment II.

Notice the small phase shift at sample point a. This is due to incomplete refocussing of the spins by the 3.84 ms z gradient pulse. This is probably due to round-off errors and the method of soft pulse generation. The theoretical change in the phase of the signal after each balanced gradient pulse is the same as that given in experiment I. The phase of the magnetization in the simulated experiment was taken to be the average of the point source magnetization phases over all transverse magnetization components greater than 1% of the thermal equilibrium magnetization value. The results of the experiment are given in Table 4.3. The simulated results are in good agreement with the theoretical calculations. Any differences can be explained by errors in the motion simulation and data sampling and storage as considered in experiment I.

4.6.3 Experiment III

In this last experiment the effect of motion on the phase of the spins during a Gaussian 90 degree RF magnetic field pulse in the presence of a linear magnetic field gradient was simulated. The purpose of this experiment was to investigate the change in the phase of the spins moving along a slice gradient during a shaped 90 degree RF pulse. The pulse sequence used as a model for this experiment is illustrated in Figure 4.9. The magnitude of both gradient pulses was $232.06 \cdot 10^{-9}$ Tesla/point. This experiment was done in two stages. In the first stage motion as shown in Figure 4.9 was modelled. After the RF pulse was finished, the motion was removed. A rephasing gradient pulse was then applied to correct for the expected dephasing of the transverse magnetization components. At this point in the experiment the phases of the transverse components were recorded for any component greater than 1% of the thermal equilibrium magnetization value. In the second stage no motion was modelled. This experiment, after the refocusing pulse was applied, gave the phase distribution of the spins positionally static along the z axis. The results of the second stage (the phase distribution along z) were subtracted from the results of the first stage on a per point basis. This gave the difference due to motion in the phase acquired by spins moving during the RF pulse. These values were averaged giving the phase of the effective transverse magnetization with respect to the y axis of the rotating frame.

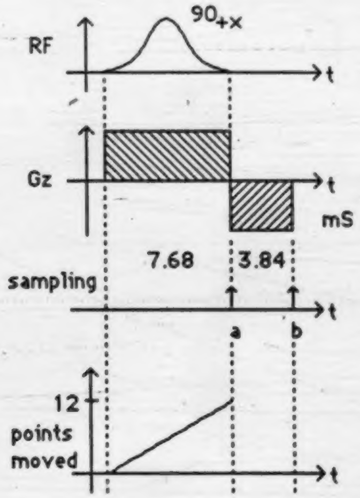


Figure 4.9 The pulse sequence modelled in experiment III.

In the case of no motion the average phase at sample point b should be zero. The phase recorded when motion was present was 13.62 degrees. Using equation 3.13) and parameters from Figure 4.8, i.e., a gradient pulse of $232.06 \cdot 10^{-9}$ Tesla/point, this phase corresponds to a gradient pulse of

$$\tau^2 = \frac{\phi}{\gamma G_z v_z}$$

$$\tau = \sqrt{\frac{\phi}{\gamma G_z v_z}}$$

$$= 1.56 \cdot 10^{-3} \text{ seconds.}$$

This result is approximately two fifths of the length of the rephasing gradient pulse. It is this factor that appears in the slice or z direction phase term in the analysis of the effect of motion on 2DFT reconstructed images in chapter 3 and Appendix A.

4.7 Conclusions and Suggestions for Improvements

As an investigative tool, the simulation package has proved to be very instructive. Not only has it provided a good understanding of NMR phenomena at the Bloch equation level, but it has shed light on the relationship between NMR signal phase and magnetization movement. Naturally, flow simulations are but one possible application for this package. Others include selective RF pulse design, the study of the effect of pulse sequences on the measurement of T_1 and T_2, investigations into the origins of imaging artifacts, and so on.

The simulation package, as it stands, is meant for low level applications. By this it is meant that only users with a good understanding of pulse program design and execution on the spectrometer can properly utilize the package. The physical functions simulated by the various modules need to be understood before a user can use the modules to emulate a pulse program. For example, to " create " a slice selective pulse sequence, the user must be aware of the relationships between shaped RF pulse duration and pulse frequency bandwidth, between tip angle and the RF pulse area, and between gradients and the Larmor frequency of the modelled magnetizations. This limits the number of potential users of the simulation.

To alleviate this restriction, the level of simulation abstraction could be raised. Writing a pseudo-pulse program compiler, similar to the compiler in place on the spectrometer, would remove the need for users to explicitly write a series of simulation module calls to emulate a pulse program. Protocols for such things as slice selection and readout gradient strength definition could be developed. These are only a few suggestions. As more people make use of the package, shortcomings will become apparent and at that time improvements could be made.

The largest problem with the simulation is the computer time required to run an experiment. At this time it is safe to say that only one dimensional experiments should be run. Even these can take many hours to finish. Perhaps a restructuring of the package away from the current coordinate system would enable it to be run using the array processor. This would mean leaving the philosophy of spectrometer emulation. As long as a user makes use of the so-called " IDEAL " modules whenever possible, the simulation run time can be kept to a reasonable amount.

5. Methods

5.1 The Flow Phantom

In all flow measurement experiments, tap water flowing anti-parallel in two glass tubes was used. Although the T_1 and T_2 values of tap water were not measured, it is safe to say that both values are long ($T_1 > 2$ seconds and $T_2 > 200$ milliseconds). Tap water was an acceptable substitute for blood since only the phase of the signal was examined. It was not necessary to model blood viscosity, relaxation rates and other such parameters, since they do not affect the phase of the NMR signal. Anti-parallel flow was accomplished by bending a single glass tube of 7.94 mm inside diameter into a U-shape (Figure 5.1a). The ends of the " U-tube " were pushed through a two-hole rubber stopper and the entire assembly seated in a one litre Erlenmeyer flask filled with tap water (Figure 5.1b). The Erlenmeyer flask served as a support for the U-tube and as a containment vessel for the non-flowing water. The " bottom " of the U-tube almost touched the bottom of the flask. The ends of the U-tube extended 12 cm past the bottom of the rubber stopper (Figure 5.1b). Two lengths of flexible tubing, approximately 3 m, were pushed over each end of the U-tube.

In an experiment, the flask was secured horizontally in a perspex tray and the entire assembly positioned in the magnet bore such that the image slice would be 2 cm from the bottom of the stopper and inside the flask. One flexible tube was connected to a 20 litre reservoir positioned 1 metre above the level of the flask and the other tube was run into a collecting bucket. Flow was regulated by a Hofstader clamp attached to the flexible tube downstream

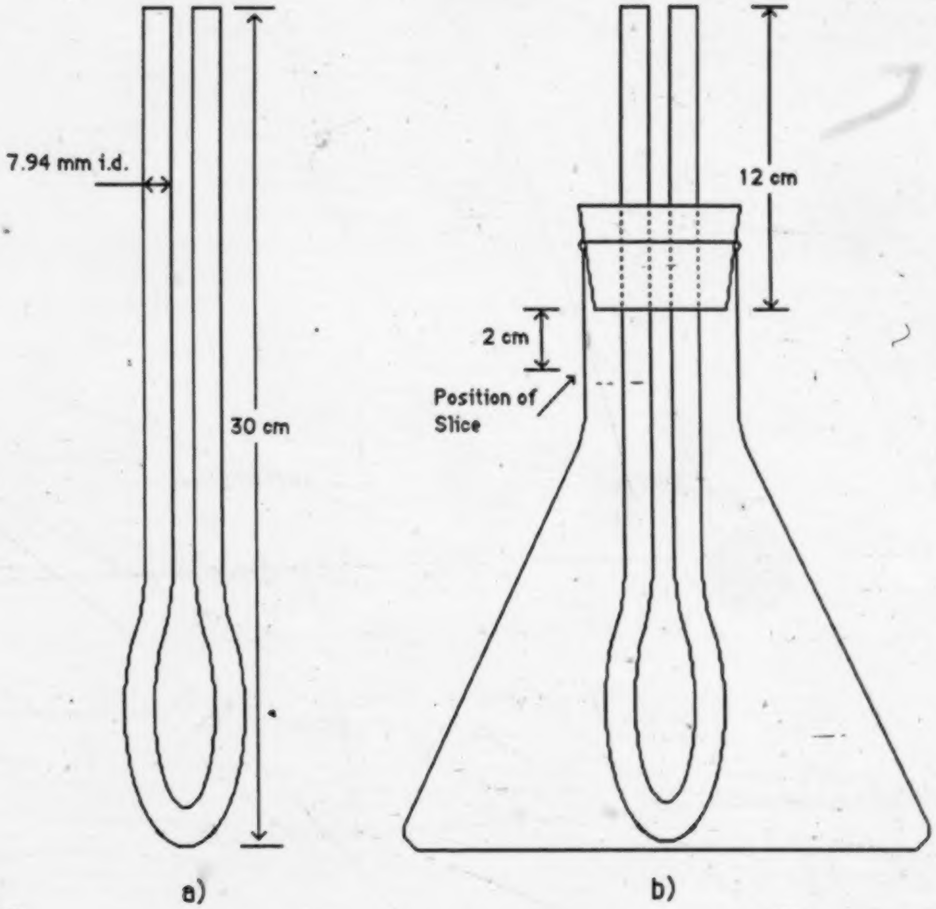

7.94 mm i.d.
30 cm
a)
12 cm
2 cm
Position of
Slice
b)

Figure 5.1 The flow phantom.

of the flask. The volume of water flow over time was measured using a graduated cylinder (200 ml) and a stop watch. The average flow rate was calculated using the equation

$$\text{rate}\left(\frac{\text{cm}}{\text{s}}\right) = \frac{\text{flow volume over time } \left(\frac{\text{ml}}{\text{s}}\right)}{\text{cross-sectional area of tube } (\text{cm}^2)}$$

$$= \frac{\text{flow volume over time } \left(\frac{\text{ml}}{\text{s}}\right)\left(\frac{1 \text{ cm}^3}{\text{ml}}\right)}{\pi\left(\frac{0.794}{2}\right)^2 \text{cm}^2} \qquad 5.1)$$

The average error in the measurement of the flow rate over the course of an imaging experiment was estimated to be less than 4%.

In all experiments constant flow velocities were maintained. Constant flow is characterized by steady laminar flow (Rouse, 1978). Laminar flow in radially symmetric tubes has a parabolic velocity profile of

$$V = 2\,V_{avg}(\,1 - b^2/r^2\,) \qquad 5.2)$$

where V_{avg} is the average flow rate as given in equation 5.1), r is the radius of the tube and b is the radial distance from the tube axis. It is easily seen that at the centre of the tube, where b = 0, the velocity will be $2V_{avg}$. In well-established laminar flow all the velocity components across the velocity profile are constant in time. As the flow rate increases, flow becomes turbulent and these components begin to fluctuate in time. The onset of turbulence can be roughly predicted using the Reynolds number, Re, which is defined as

$$Re = \frac{\rho V_{avg} D}{\mu} \qquad 5.3)$$

where D is the tube diameter, μ is the viscosity of the liquid and ρ its density.

The period of laminar flow is described in McDonald (1974) as falling into three regimes. In the first regime there is completely undisturbed flow with streamlines (laminae) running parallel with the tube walls. At higher velocities (higher Reynolds numbers) these streams begin to exhibit a wavy motion. This is the second regime. As velocity increases and the third regime is entered, these oscillations grow and vortices begin to form at any borders in the tube (e.g., where the tube is spliced or bent or leaves the reservoir) and are carried down the tube. As vortices become larger and more frequent, this regime merges into turbulent flow. A Reynolds number of 1600 is usually ascribed to the onset of the third regime and 2000 to the onset of turbulent flow. Hence, to ensure flow within the second regime, flow rates were kept such that the Reynolds number was almost always less than 800. For a tube of 0.794 cm i.d. and room temperature water, this corresponded to

$$\begin{aligned} V_{avg} &\le \frac{Re\ \mu_{H_2O}}{\rho_{H_2O}\ D} \\ &\le \frac{800 \cdot 0.01\ \text{gm cm}^{-1}\text{s}^{-1}}{1\ \text{gm cm}^{-3} \cdot 0.794\ \text{cm}} \\ &\le 10\ \text{cm s}^{-1} \end{aligned}$$

5.2 NMR Imaging Techniques

All NMR measurements were made with a Bruker CXP spectrometer and a 40 cm horizontal bore superconducting magnet operating at 2.35 Tesla. A 100 MHz (γ is 100.19 MHz/2.35 Tesla for protons) slotted resonator imaging coil was used to transmit radiofrequency electromagnetic radiation to the sample and to detect NMR signals from the sample. Signals were sampled at 41.67 kHz. The time between scans was always 1 second. The delay between the 90° RF pulse and the spin echo was 34.538 ms. In all imaging experiments four echoes were generated and acquired. Two averages were taken for each multi-echo set.

A typical imaging experiment begins with the placement of the object in the magnet bore. The point in the sample corresponding to the centre of the desired imaging slice is positioned 57.3 cm from the front plate of the magnet. This corresponds to the region of greatest magnetic field homogeneity. A standard spectrometer preparation protocol or command file which allows the user to prepare the spectrometer for an experiment (i.e., tune and match the imaging coil, shim the static magnetic field, set the 90 and 180 degree RF pulses, etc.) is then invoked from the console. At the end of this sequence the experiment to be run is chosen from a software menu. In the case of flow measurements, the IMFL (IMaging of FLow) command file is invoked (Appendix F). This file reads in the necessary values of various spectrometer parameters. At the completetion of IMFL, the user is prompted to invoke the IMFLPL (IMaging of FLow, pulse Program Load. Appendix G) command file. The flow measurement pulse program is read in and a routine to interactively adjust the amplitude of the Gaussian shaped 90 degree RF pulse is started. This is a standard procedure whereby all gradient switching is disabled, a gradient constant in time is established along the direction normal to the slice (in most cases along the z axis) and the pulse program is run. The result is a spin echo signal from the magnetization within the slice. The gain of the shaped RF pulse is adjusted until an optimum echo amplitude is attained. IMFLPL differs from the usual pulse program loading routine in that the user is allowed to specify the strength of the time-constant z gradient. At the completion of IMFLPL the user is requested to select a gradient set (x, y and z gradient files) to be used in generating the image data. For a transverse image (i.e., a slice normal to the z direction), each x and y gradient set defines an imaging " magnification " factor. That is, the strengths of the gradients are set such that an object of a given diameter will occupy a certain width in the image. For example, consider a spin echo imaging sequence using a readout, or x, gradient of $48.96 \cdot 10^{-6}$ Tesla cm^{-1} during the acquistion period. As mentioned, the spin echo is sampled at 41.67 kHz. Recalling the Nyquist criteria, this sampling rate corresponds to a spectral window of 20.83 kHz. In section 2.8 it was shown that a linear magnetic field gradient gave a linear variation in Larmor frequency along the gradient direction. Thus, the readout gradient corresponds to a 2083 Hz cm^{-1} change along the x axis. It is clear that the sampling window corresponds to a width of 20.83 kHz / 2083 H cm^{-1} = 10 cm. Thus, the width of the imaging slice in the x direction must be less than 10 cm in order that the slice be completely within the image (at least in the readout direction).

After the gradient set is established, the command file IMFLG (IMaging of FLow, Go. Appendix H) is invoked in order to set the attenuation of the acquired signal and to check the " profile " of the image. A profile is simply the Fourier transformation of a spin echo acquired with no phase encoding. The result is a projection of the spin density in isochromatic planes normal to the frequency encoding, or readout, direction onto the frequency encoding axis.

At the completion of the IMFLG command file there is an optional step. If this is the beginning of an imaging session, it is wise to ensure that the echo is centred in the sampling window (refer to section 3.1). This is done by invoking IMFLTR (IMaging of FLow echo TRim. Appendix I). In this routine, the readout direction dephasing gradient pulse amplitude is adjusted to centre the echo. Each time the gradient pulse is altered, a scan is made with no phase encoding gradient on. The spin echo acquired is Fourier transformed to give a complex profile. The phase at each point across this profile is summed and assigned to a variable, 1PHZ (Bruker notation). Ideally 1PHZ should be zero. Practically, any absolute sum less than 100 is acceptable.

The final step in the imaging experiment is the invocation of IMFL1 (IMaging of FLow, 128 phase encoding steps. Appendix J) which queries the user for a data file name and then proceeds with the imaging experiment.

5.3 Image Reconstruction and Data Analysis

All image reconstruction and data analysis was done using Digital Corp. VAX 11/750 computers. The generation of phase images requires the full two dimensional complex Fourier transformed image data set. The image reconstruction facility the Bruker provides will only produce modulus images, hence a separate reconstruction program had to be developed and implemented. Time domain data (the spin echoes from N_y scans) are transferred from the CXP via a parallel link to the VAX. The user has then the option of running a phase correction program or else proceeding directly with the reconstruction program.

The phase correction program, PHASCOR (Appendix K), allows the user to visually inspect the frequency domain phase of the spin echo. The user can correct for discrete shifts in time of the spin echo (refer to section 3.1) by specifying the number of sample points to move the echo left or right in the sample window. These corrections are saved in a file for use by the reconstruction program.

The reconstruction program, RECO (Appendix L), reads the raw time domain data file into main memory, shifts the echoes in their sample windows if requested, and passes the data to an array processor (Computer Design and Applications Inc., MSP-3000). The array processor performs the complex two dimensional Fourier transformation and passes the resultant complex data set back to RECO. The complex data is stored in a file which is used by the image generation program.

Both PHASCOR and RECO have routines to " unscramble " the data from the 24 bit Bruker word format to the VAX 32 bit word format. The basic subroutines in both programs (Appendices K and L) were written by Dan Gheorghiu of the Department of Applied Sciences in Medicine. These routines make use of either the array processor or, if it is busy, the VAX.

The final step in image reconstruction, the generation of images, is performed by the program IMGMAKE (Appendix M). This program queries the user for information about the experiment and that information is saved in " header " data blocks. These blocks are written at the beginning of each image file. Next the user specifies the type of image desired, the choices being, real, imaginary, modulus, or phase (as defined in section 2.8). The " raw " complex data file (generated by RECO) is then read in and the images are calculated and saved to disk. This completes the image reconstruction step.

The program PHASME (Appendix N) facilitates the measurement of the average phase in rectangular areas defined by the user within the image. The images are displayed on a graphics monitor and the regions of interest are outlined by the user with a track ball accessory. For multi-echo images or multi-image sets that show the same position of the subject it is necessary only to define the regions of interest in the first image. The information about these regions is saved and, as successive images are read in and displayed by PHASME, the region information is recalled and the same regions of interest are defined for the current image. Results of these measurements are saved to a file for later output to a print device.

As an example, consider four images made of the U-tube phantom (section 5.1) and that it was not moved between the image acquisitions. A typical image is represented in Figure 5.2.

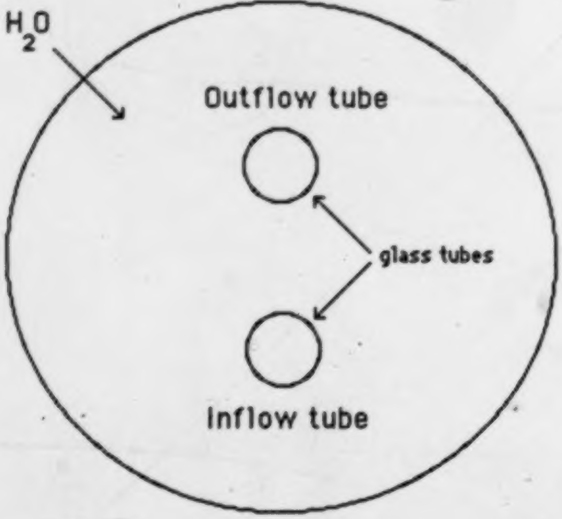


Figure 5.2 A sketch of a typical image of the U-tube phantom.

The objective of the flow measurement experiment is to measure the difference in phase between the non-moving water in the flask and the flowing water in the tubes. Hence regions within the flask and within the tubes must be defined. One arrangement of regions is illustrated in Figure 5.3. The average of the values in regions one through four inclusive is taken to be the average phase around the outflow tube. This value subtracted from the measured phase in region five gives the difference in phase between the static water and water flowing in the outflow tube. The same can be done for the inflow tube. In all flow measurement experiments reported in Chapter 6, regions 5 and 10 were very small (2 X 2 pixels) and in the centre of the tubes. This corresponded to the region of peak flow veleocity (see section 5.1). These measurements are repeated automatically for each of the remaining three images subsequently read in by PHASME.

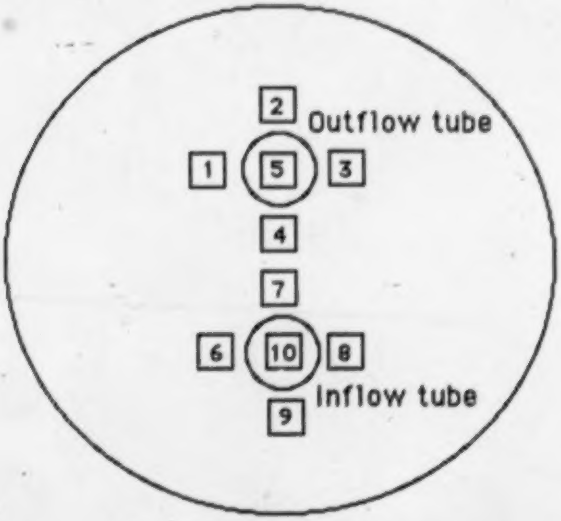


Figure 5.3 A set of region definitions used to measure the average phase of the static water around a tube and the phase of the water flowing in the tube.

6. Results and Discussion

As mentioned in the Introduction, the purpose of this project was to understand the factors affecting the phase of 2DFT images generated by spin echo pulse programs. After an initial failure to measure fluid flow in a phantom and the realization that there were indeed significant obstacles to overcome, the pursuit of a flow measurement technique became a series of " learning " experiments. Each experiment has provided information about how the phase of the NMR signal behaves in a NMR experiment. From what was learned, progress has been made towards a method that utilizes the difference between the phase of the signal from non-moving spins and the phase of moving spins as a measure of the fluid velocity (refer to section 3.2).

In an attempt to structure the results obtained over the past two years, the chapter has been subdivided into sections that deal with distinct, yet inter-related aspects of the flow measurement technique development. In all the NMR images presented in this chapter, the frequency encoding axis is vertical and the phase encoding axis is horizontal. The usual pixel size was 0.8 mm in the phase encoding direction by 0.4 mm in the frequency encoding direction.

6.1 Phase Variations

Preliminary attempts at implementing an imaging pulse program revealed unacceptable (i.e., greater than 20 degrees) variations in phase across an image of non-moving water (Figure 6.1). These variations were manifest predominately in the frequency, or x, direction. The variation in the

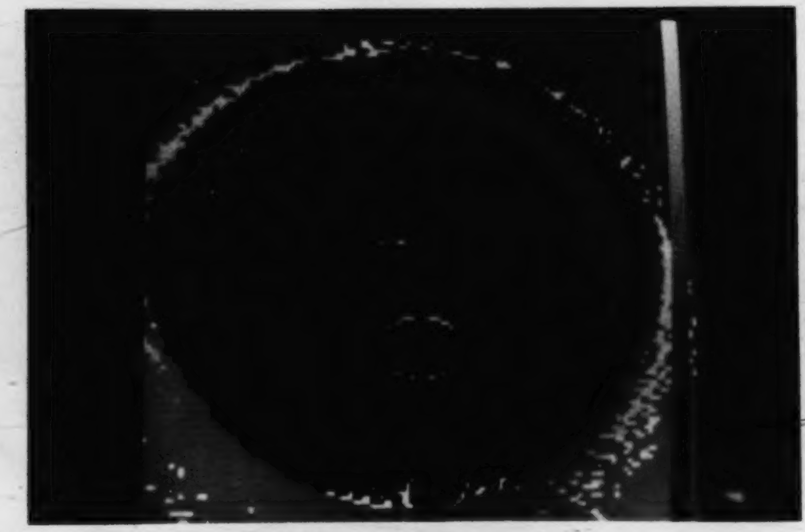


Figure 6.1 A phase image showing smooth variations along the frequency or x axis (vertical) and the phase or y axis (horizontal) encoding directions. On the grey scale, phase varies linearly between π (white) and -π (black).

phase encoding, or y, direction was not as large, but still apparent (Figures 6.2a and 6.2b, respectively). This suggested that these variations were related to the frequency and phase encoding gradient pulses of the imaging pulse program. Since, for a given imaging sequence, the number and size of the phase encoding gradient pulses are fixed, only the effect of the frequency, or readout, gradient pulses on the phase of the image could be examined. Inspection of a typical spin echo imaging pulse program (e.g., Figure 2.19) reveals that the dephasing of the spins by the x gradient pulse between the 90 and the 180 degree RF pulses must be reversed by the readout gradient pulse that follows the 180 degree pulse in order that a spin echo be formed. By adjusting the area (duration multiplied by the amplitude) of the first gradient pulse, the spin echo may be shifted in the sampling window. If the centre of the spin echo is not at the centre of the sampling window, it is said to be shifted in time (the time origin being at the centre of the sampling window). A shift of the spin echo signal in the time domain results in, after the Fourier transformation of the signal, a non-zero phase distribution in the frequency domain (see section 3.1). The frequency encoding gradient pulses do not vary from scan to scan in the imaging sequence, therefore, any time shift of the spin echo will be present in each scan. The result in the final image will be a linear variation in phase across the image along the frequency encoding direction. An image from a badly time-shifted spin echo is shown in Figure 6.3a, and an image from the spin echo centred in the sampling window is shown in Figure 6.3b. Notice that there remains a small variation in phase along the frequency encoding direction in Figure 6.3b due to the non-continuous nature of the gradient pulse amplitude generation (refer to section 3.1).

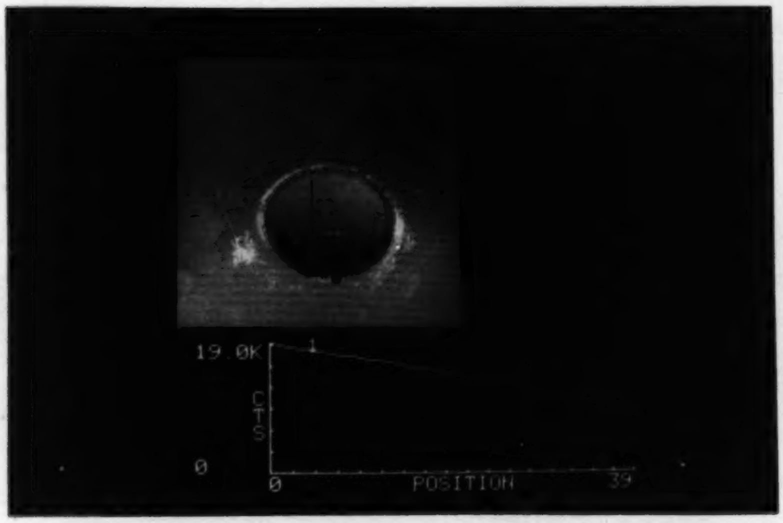


Figure 6.2 a) A profile of the variation in phase along the frequency encoding direction. The dependent axis of the graph gives measured pixel intensities. The range is [0,32767] with 0 representing $-\pi$ and 32767 representing π.

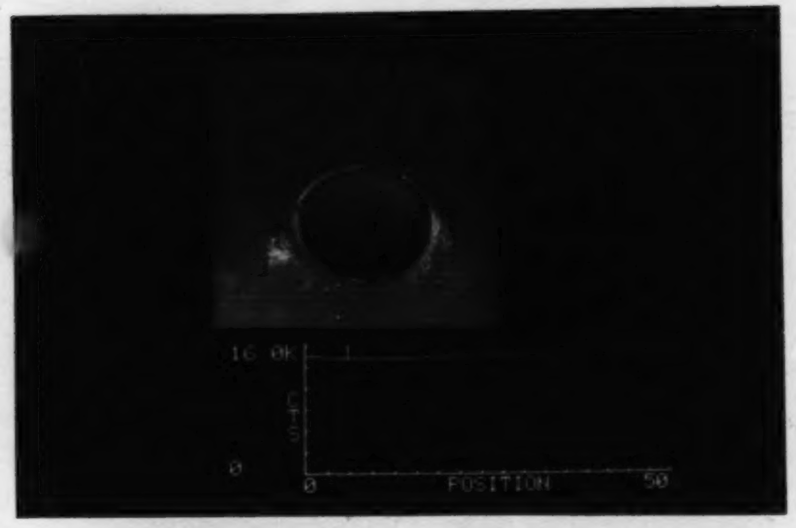

16.0K
CTS
0
0
POSITION
50

Figure 6.2 b) A profile of the variation in phase along the phase encoding direction.

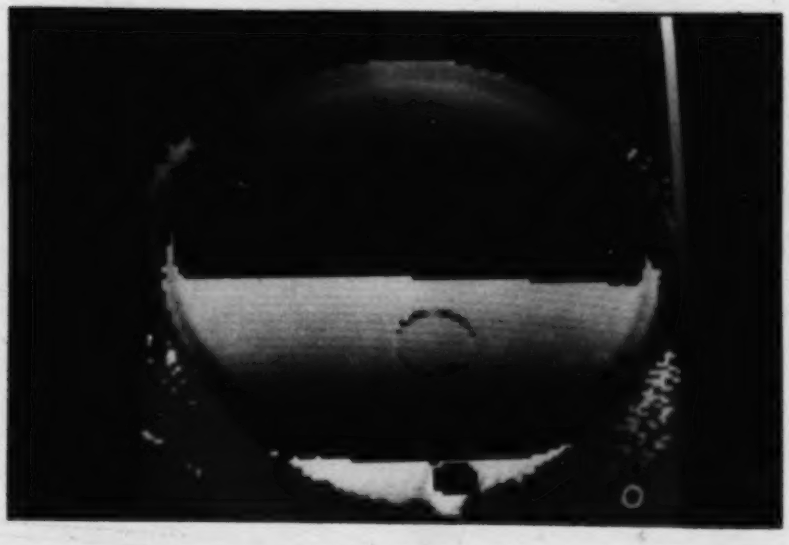


Figure 6.3 a) The effect of a badly mis-centred spin echo on the phase of an image (the frequency encoding direction is vertical).

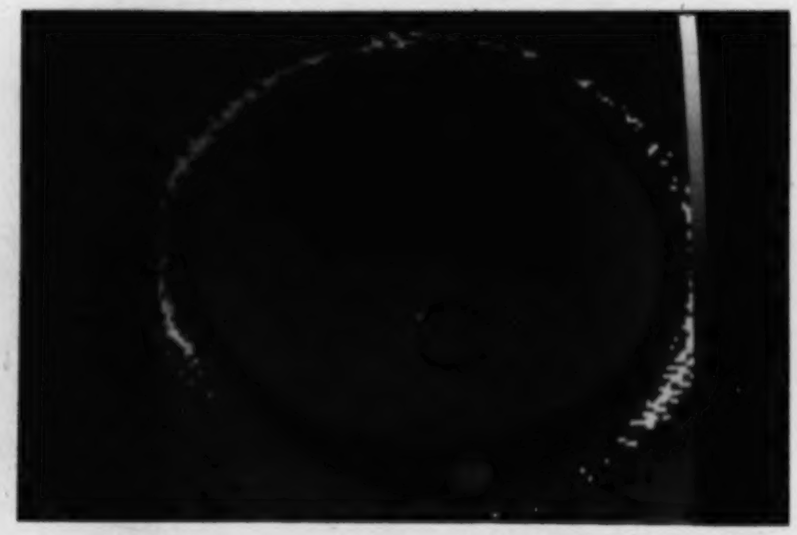


Figure 6.3 b) The appearance of a phase image when the spin echo is centred in the sampling window.

As a result of these investigations, a routine to interactively centre the spin echo in the sampling window was later developed and made a part of the imaging preparation sequence (see section 5.2, the command file IMFLTR). After the correction for the phase error in the frequency encoding direction as caused by discrete shifts in time of the spin echo (see section 3.1), there still remained small, smooth phase variations in both directions across the image. The effect of these variations in the measurement of the phase change due to flow in the image (see analysis method in section 5.3) was unknown. It was felt that, if these variations were severe enough, a method of phase correction such as suggested by O'Donnell (1985) could be implemented. O'Donnell's technique produces two phase images, one phase sensitive to flow and one not. Both images show the same phase for non-moving spins. Hence, when one is compared to the other, only the difference in phase from the flowing spins remains. The non-moving spin phases cancel. If, on the other hand, the variations in phase that remained were not detrimental, only one image need be produced, thus saving time.

In addition to the " image-wide " variations in phase that were investigated, phase variations near the glass tubes and the glass walls of the flask were observed. These are probably due to a change in the local magnetic field resulting from the difference in the magnetic susceptibilities of water, glass and air. An imaging experiment demonstrating the severity of this " susceptibility artifact " was performed using a glass tube mounted in a flat perspex slab (Figure 6.4). An image was made with the tube in the x-y plane and in the centre of the slice. At the interface between the perspex and the glass the static magnetic field was altered to the extent that the hydrogen nuclei in that region were off resonance and not rotated 90 degrees by the selective 90 degree RF pulse. This is shown by the lack of signal in the central portion of the modulus image in Figure 6.5a. When the slice was shifted (by shifting the RF transmitter frequency) 3 mm past the tube, a second image revealed the shifted central portion of the water in the tube (Figure 6.5b).

There is no correcting for this artifact as any material will cause a change in the magnetic field. Thus, all measurements of the image phase had to be made far from the tube and the flask boundaries.

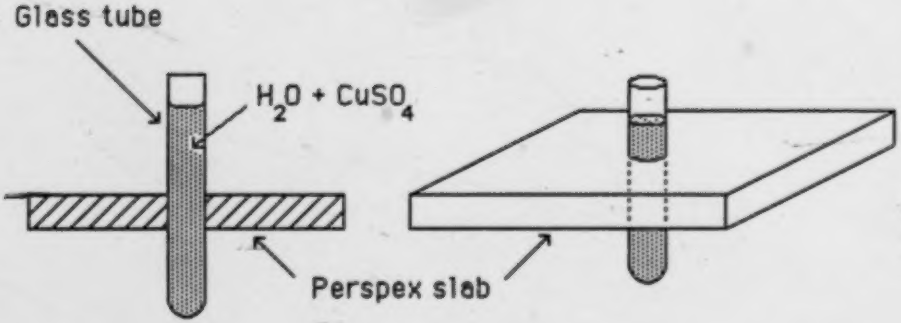


Figure 6.4 A glass tube, 4 mm i.d., of $CuSO_4$ doped water supported in a perspex slab.

Figure 6.5 a) A modulus image of the tube centred in the imaging slice. On the grey scale, black is the lowest, or zero signal intensity and white is the highest.

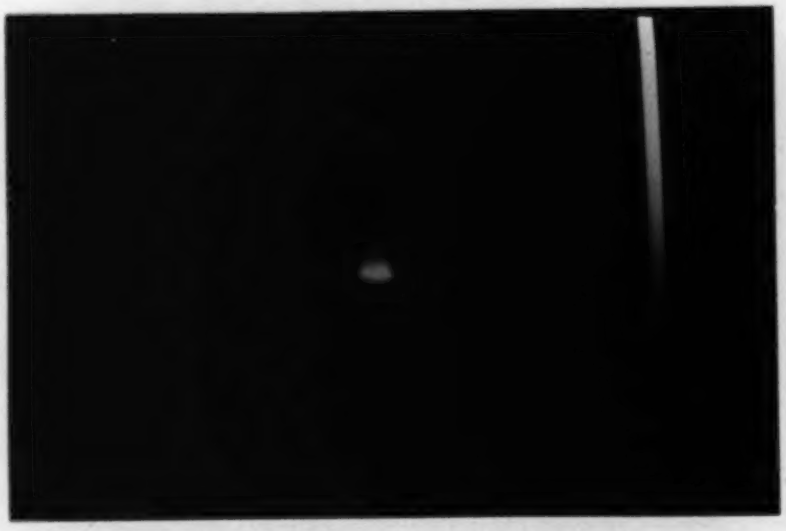


Figure 6.5 b) A modulus image of the slice now shifted 3 mm from the centre of the tube. The " missing " water is now excited by the 90° RF pulse.

6.2 Dependence of Flow Measurement Range on Slice Selection Sequences

Having established a method to remove the largest variation in the phase of an image, i.e., variations in the frequency encoding direction due to spin echo mis-centring, flow induced phase differences were re-examined. At this time is was decided to concentrate on single-echo spin echo images. The pulse program at that time corresponded to the one shown in Figure 6.6. Gradient amplitude values are given for only the z direction magnetic field gradient pulses since flow was in that direction only (recall that x and y direction gradients will affect the phase of the signal only when there are flow components in those directions. See section 3.2).

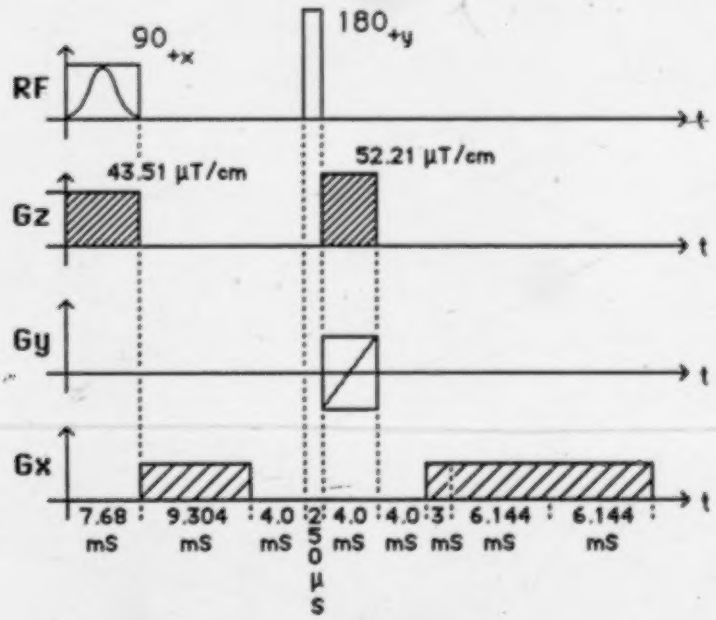

RF
90_{+x}
180_{+y}
t
Gz
43.51 µT/cm
52.21 µT/cm
t
Gy
t
Gx
t
7.68 mS
9.304 mS
4.0 mS
250 µS
4.0 mS
4.0 mS
3 mS
6.144 mS
6.144 mS

Figure 6.6 A pulse program used to study the effect of the slice selection sequence on the phase of the NMR signal from flowing liquid.

There are no special flow phase-encoding gradient pulses in this pulse program. It was desired to investigate the effect of the slice selection sequence on the signal phase from flowing spins.

An experiment began with the set up of the U-tube flow phantom (section 5.1) and preparation of the spectrometer (including execution of the spin echo centring routine, IMFLTR). Then, for a series of different flow rates, phase images were acquired. Measurements were made of the difference between the phase of the flowing fluid region (inside the tubes) and the phase of the non-flowing fluid for each image generated. These phase angles were plotted against their corresponding flow rates and this data was fitted to a straight line. Again, it should be noted that the region measured in the tubes corresponded to the region of peak flow rate (i.e., small regions in the very centre of the tube where b, in equation 5.2, is very small and, therefore, $V \approx 2V_{avg}$).

The slope values calculated from these initial experiments were between 25 and 30 degrees per cm s^{-1}. If physiological flow rates were to be measured (0 to 150 cm s^{-1}), then these slopes were clearly too large. That is, it would take a flow rate of only 6 cm s^{-1} to reach the limit, 180 degrees, of the phase encoding range (after 180 degrees the phase would " wrap " and start increasing from -180 degrees).

It was recognized that the source of this large phase shift at low velocities was the slice defining gradient pulse sequence. It was known that the phase shift due to constant flow is proportional to the strength of the gradient (equation 3.13). This suggested lowering the amplitudes of the slice selection gradient pulses. However, lowering the strength of the slice defining gradient means that a thicker slice will be selected (i.e., changing $\underline{G}_r$ in equation 2.55 changes $\underline{B}_{eff}$. Refer to the discussion in section 2.9). In an animal or a human there are only a few vessels that run straight for any great distance. The thicker the slice is, the less likely that a vessel will run perpendicular to the slice for the entire width of the slice. If the vessel is not perpendicular to the slice, then the assumption of flow with velocity components only along the z axis is no longer valid. Thus, there had to be some trade-off between slice thickness and flow velocity phase encoding. A simple two part experiment was devised to measure slice thicknesses at various z direction magnetic field gradient strengths. In the first part, two spheres of 5 mm diameter were filled with water and centred 4 cm apart along the z axis of the magnet. A z gradient was turned on and allowed to " settle ". Then the samples were irradiated by a 90 degree pulse and the resulting FID recorded. Following Fourier transformation of the FID, the separation in Hz of the signals from each sphere was measured. Multiplying this quantity by $2\pi/\gamma$ and dividing by 4 cm gave the strength of the gradient for that particular gradient setting in Tesla per cm. The Bruker CXP spectrometer gradient strengths are related to an integer variable which has the domain [-1800, 1800] and which is referred to as the "gradient IOD" value.

After several measurements at various IOD values (e.g., -500, -300, -100, 0, 100, ...), a scalar factor was found relating gradient strength to IOD value. This factor was $87.0205 \cdot 10^{-9}$ Tesla cm^{-1} IOD^{-1}. For example, an IOD value of 500 is equivalent to a gradient strength of $43.51 \cdot 10^{-6}$ Tesla cm^{-1}.

The second part of this experiment measured the frequency bandwidth of the slice selective 90 degree RF pulse. A flask 10 cm in diameter, filled with water, was placed in the magnet and irradiated by the following spin echo pulse sequence; selective $90°_{+x}$ - τ_1 - $180°_{+y}$ - τ_2 - acquire(spin echo). The selective 90 degree RF pulse was Gaussian in shape and 7.68 milliseconds long. The non-selective 180 degree RF pulse was rectangular and 250 microseconds long. τ_1 was 13.304 milliseconds and τ_2 was 11.000 milliseconds. The signal was sampled every 24 microseconds giving an effective spectral resolution of 41 Hz. During the entire sequence a gradient in the z direction was present, but its value is unimportant in this experiment. After the acquired spin echo signal was Fourier transformed, the half-height width of the resulting " slice profile " (refer to section 2.8) was measured. It was found to be 492 Hz or about 500 Hz. This was taken to be the width , S_{BW}, in Hz of the slice. Over this frequency band the magnetization is considered to be rotated 90 degrees by a 90 degree RF pulse.

From the results in part one and two an equation is easily derived for the slice thickness. Slice width, SW, in cm is given by

$$SW = \frac{2\pi S_{BW}}{\gamma G_z} \tag{6.1)}$$

where G_z is the z direction magnetic field gradient strength in Tesla cm^{-1}. Slice widths for various G_z values are given in Table 6.1. From Table 6.1 it is seen that the early flow measurement experiments were done using a slice selection gradient strength ($43.51 \cdot 10^{-6}$ Tesla cm^{-1}) that resulted in a 3 mm slice. It was felt that a slice of up to 15 mm could be used in flow measurement experiments looking at flow in large vessels (> 3 mm). Use of slice thicknesses from 10 to even 20 mm is common in the large bore clinical NMR imaging systems of today.

IOD value	G_z (Tesla cm^{-1})	Slice width (cm)
10	$870.2 \cdot 10^{-9}$	13.5
50	$4.35 \cdot 10^{-6}$	2.7
90	$7.83 \cdot 10^{-6}$	1.5
135	$11.75 \cdot 10^{-6}$	1.0
270	$23.49 \cdot 10^{-6}$	0.5
500	$43.51 \cdot 10^{-6}$	0.27

Table 6.1 Slice thicknesses for various gradient settings and a Gaussian shaped 90 degree RF pulse of 7.68 ms duration.

However, repeating the earlier flow measurement spin echo imaging experiments and phase analysis, it was found that a gradient strength of $7.83 \cdot 10^{-6}$ Tesla cm^{-1} still gave a slope of 6.5 degrees per cm s^{-1}. This was still unacceptable as the maximum flow rate measureable was only $180/6.5 \approx 27$ cm s^{-1}.

Having reached an upper limit on the thickness of the slice, it was decided to try and change the slice selection sequence itself. The delay between the slice selection gradient pulse during the selective RF irradiation and the slice refocussing gradient pulse after the 180 degree RF pulse was recognized as being a factor contributing to the phase encoded for moving spins. This delay allows flowing transverse magnetization to move along the z direction to a position of much different magentic field strength by the time the refocussing gradient pulse is applied. The magnetization then precesses at a much different rate than when the selective RF pulse ended. This difference in rate is manifest in the phase of the signal. The way to eliminate this delay is to apply the refocussing gradient pulse immediately after the selective RF pulse, but in the direction opposite to that of the slice selection gradient pulse. This scheme was discussed in section 2.9 and illustrated in Figure 2.18.

Such a slice selection sequence was implemented. However, this sequence would not produce acceptable images, i.e., severe artifacts were produced in both the phase and the modulus images. Since the only changes were in the temporal location and direction of the refocussing gradient pulse, the problem was related to this gradient pulse. Three experiments were run to try and shed some light on the problem.

In the first experiment the linearity of the z direction magnetic field gradient was checked. Eleven tubes of 4 mm inside diameter were filled with water and placed at 1 cm intervals along the z axis at the radial centre of the magnet. The centre tube was positioned at 57.3 cm from the front plate of the magnet. A simple spin echo pulse sequence, $90°_{+x}$ - τ_1 - $180°_{+y}$ - τ_2 - acquire(spin echo), was used. A gradient was applied during the acquisition period causing the magnetization in each tube to precess at a frequency proportional to the strength of the magnetic field at that spatial location. When the acquired spin echo was Fourier transformed, the result was eleven separate peaks in the frequency spectrum whose separation in frequency depended on the strength and linearity of the magnetic field gradient. This experiment was performed for IOD values of -500, -250, 250, and 500. The objective was to determine the linearity of the magnetic field gradients along the positive and negative z directions and to see whether or not they had a common null point (i.e., the point in the magnetic field where only $\underline{B}_0$ is present).

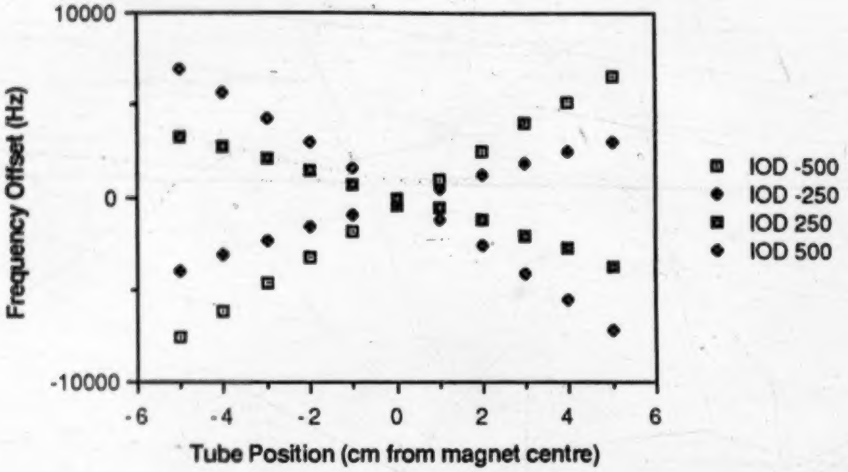


Figure 6.7 Results of the z direction gradient linearity experiment. An IOD value of 250 corresponds to a gradient of $21.76 \cdot 10^{-6}$ Tesla/cm. An IOD value of 500 corresponds to a gradient of $43.51 \cdot 10^{-6}$ Tesla/cm. The sign of the IOD value indicates the z direction of the gradient.

The results are plotted in Figure 6.7. It shows good magnetic field gradient linearity for both z direction gradients and that they have a common null point.

The next experiment tested the radial symmetry of the z direction magnetic field gradient. A disk phantom of diameter 8 cm and thickness 3 mm was constructed. The phantom was positioned at the centre of the magnet such that it was perpendicular to the z direction. It was fixed to a piece of styrofoam about 5 cm thick which was in turn fixed to a perspex disk. This perspex disk fit snugly inside the bore of the imaging coil. To the other side of the perspex disk a long rod was affixed so that the entire assembly could be shifted along the bore of the magnet. This arrangement is illustrated in Figure 6.8.

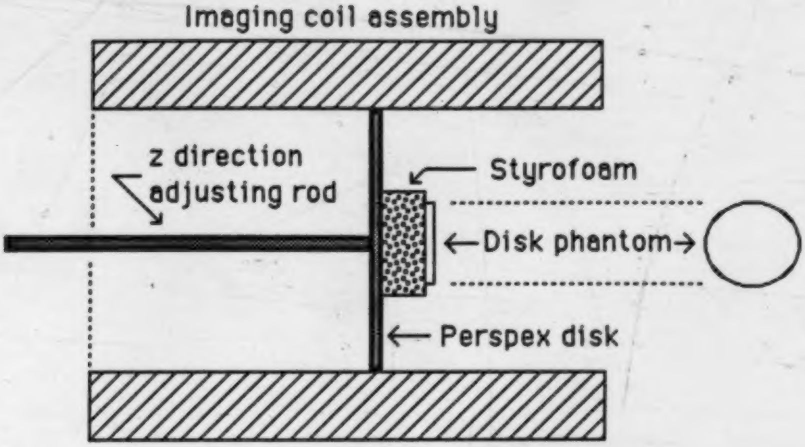

Imaging coil assembly
z direction adjusting rod
Styrofoam
Disk phantom
Perspex disk

Figure 6.8 The experimental setup of the 3 mm thick disk phantom.

The styrofoam, being mostly air, minimized the effect of the magnetic susceptibility of the perspex disk on the magnetic field near the disk phantom.

The disk phantom was filled with copper sulfate doped water (< 5mM) and positioned at 57.3 cm from the front plate of the magnet. Images, using a standard spin echo imaging sequence, were generated at 56.7, 57.0, 57.3, 57.6, 57.9, and 58.2 cm from the front plate. Some results are shown in Figure 6.9.

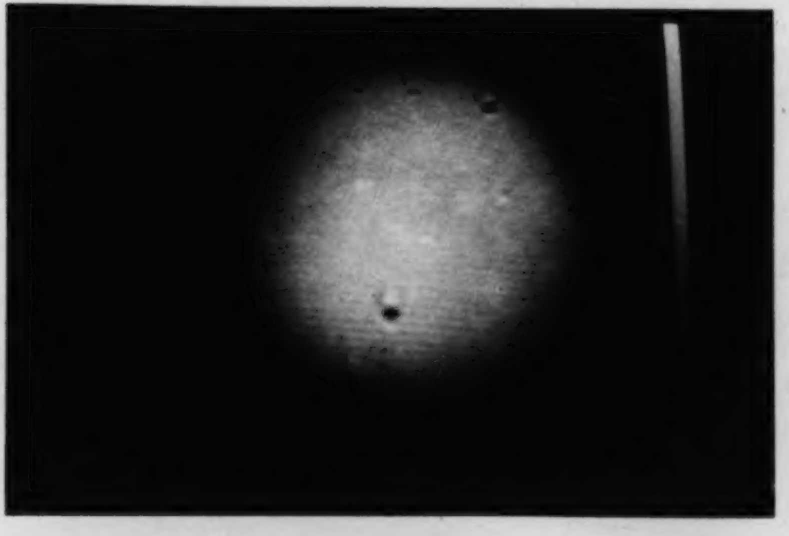


Figure 6.9 a) A modulus image of the disk phantom at 57.0 cm from the front plate of the magnet.

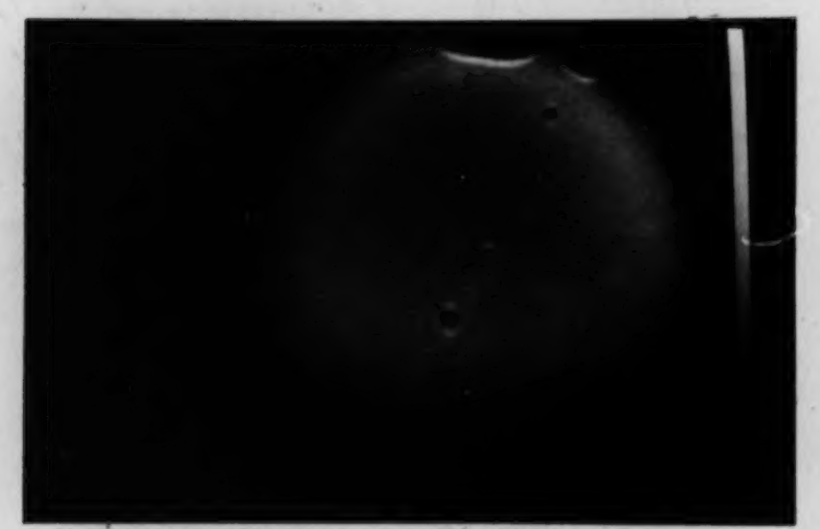


Figure 6.9 b) A modulus image of the disk phantom at 57.3 cm from the front plate of the magnet.

Figure 6.9 c) A modulus image of the disk phantom at 57.6 cm from the front plate of the magnet.

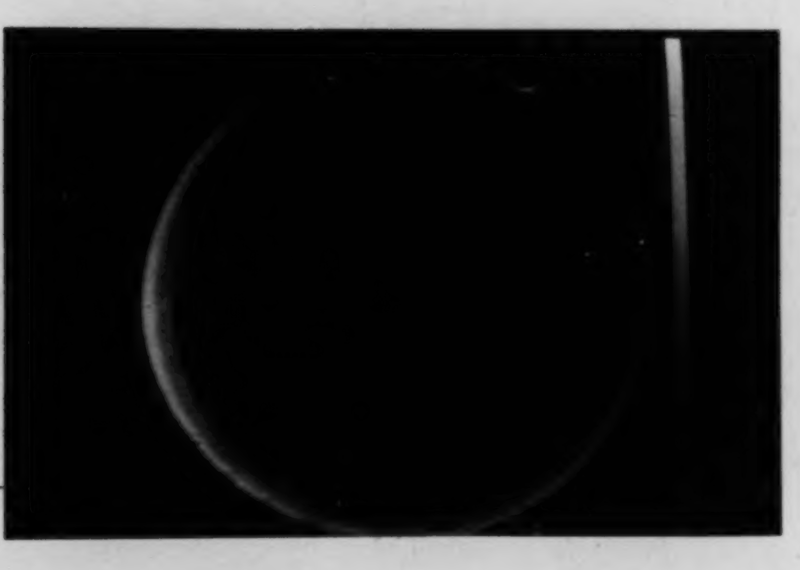


Figure 6.9 d) A modulus image of the disk phantom at 57.9 cm from the front plate of the magnet.

Since the selective 90 degree RF pulse was Gaussian shaped, a Gaussian shaped distribution along the z direction of tip angles would be expected across the slice. The intensity of the NMR signal would be greatest for a 90 degree tip angle and decrease with decreasing tip angle. Thus, the less intense regions in Figure 6.9 reflect tip angles less than 90 degrees. The regions of greatest signal intensity can be seen to " move " radially from image to image. Since each image represents a different position along the z axis, the movement of the high intensity signal regions traces the 90 degree tip angle positions in space. In other words, there is a radial dependence of the z direction magnetic field gradient. Application of a gradient along the z direction can be thought to produce curved iso-magnetic field surfaces, the centres of which are normal to the z axis. The result, in a " three dimensional object ", would be the selection of a curved slice as illustrated in Figure 6.10.

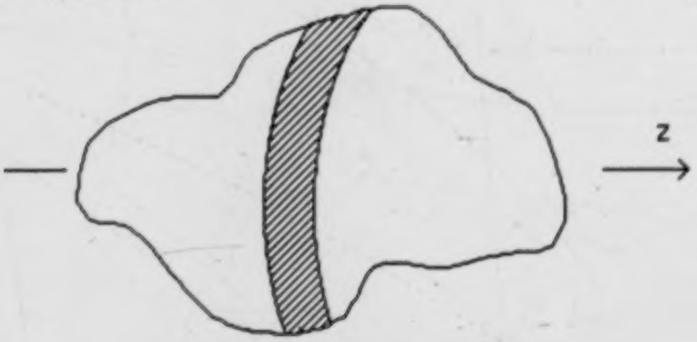


Figure 6.10 A curved slice that is the result of the radial dependence of the z direction magnetic field gradient.

When the direction of the magnetic field gradient is reversed, this radial dependence inverts, i.e., the iso-magnetic field surfaces are curved in the opposite direction. If a gradient pulse in the opposite direction is used, the result would be similar to superimposing the two oppositely curved surfaces (Figure 6.11)

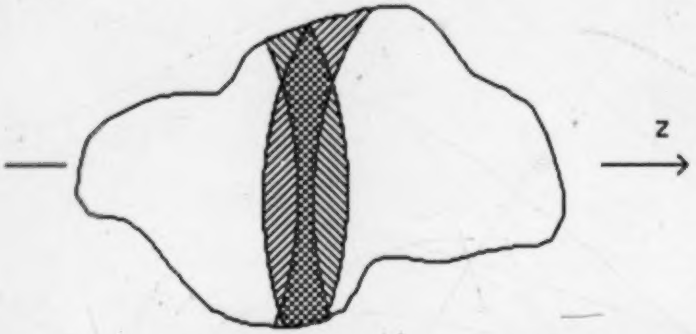


Figure 6.11 A visualization of the effect of a slice selection gradient pulse along one direction and a slice refocussing gradient pulse along the opposite direction. Only the spins in the double hatched areas will come close to refocussing.

Only the volume indicated by the double hatching would be properly refocussed. This would most likely produce distortions in the image seen as regions of diminished signal.

The third experiment involved the measurement of the magnetic field gradients produced by eddy currents in the cryostat that are induced by gradient switching. The experimental procedure for gradient pulses along the z direction was as follows. Two 15 mm spheres of water were placed 4 cm either side of the centre of the magnet (57.3 cm from the front plate) along the z axis. The pulse program consisted of a set of gradient pulses followed by a delay τ, then a 90 degree RF pulse and the acquisition of the FID. Eddy currents generated in the cryostat will result in a small magnetic field gradient which changes the effective magnetic field at each sphere. If this gradient were zero, the FID would consist of only one frequency component and its frequency spectrum would contain only one peak. If the gradient is not zero, the FID would consist of two frequency components and the spectrum would contain two peaks whose separation would depend on the strength of the gradient. Four different gradient pulse sets were tested. The separation of the spectral peaks from the two spheres is plotted for each gradient pulse set as a function of the delay time τ. The four gradient pulse sets are labled α, β, δ, and ζ and are illustrated in Figure 6.12

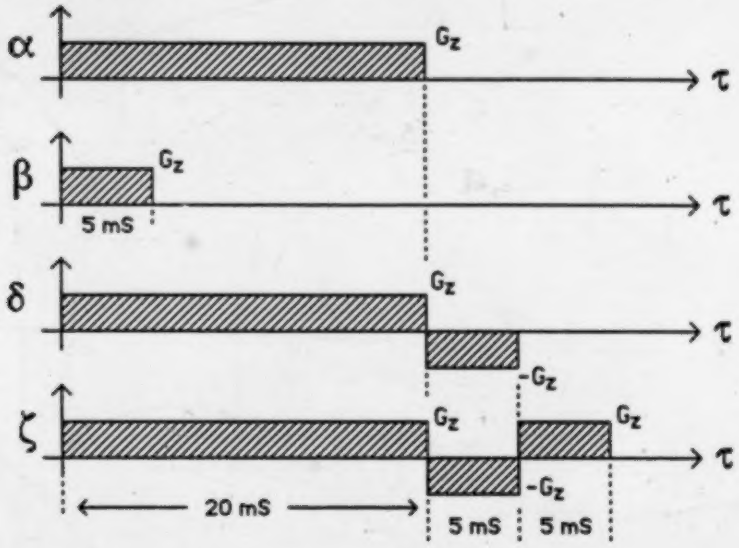


Figure 6.12 The four gradient pulse sets. All gradient amplitudes are $43.51 \cdot 10^{-6}$ Tesla cm^{-1} and the pulses are along the positive or negative z axis as indicated.

The change in the Larmor frequencies of the spheres after a gradient pulse set is plotted in Figure 6.13. In each case it is clear that the magnetic field is not static until a very long time (in terms of an imaging sequence) after the gradient pulse is zero. If a typical frequency encoding gradient strength of 2000 Hz cm^{-1} ($47 \cdot 10^{-6}$ Tesla cm^{-1}) is considered, the values depicted in Figure 6.13 represent an error of about 1%. However, it must be remembered that these eddy current-generated magnetic field gradients are present for a long time. Thus, they can produce a large cumulative phase error.

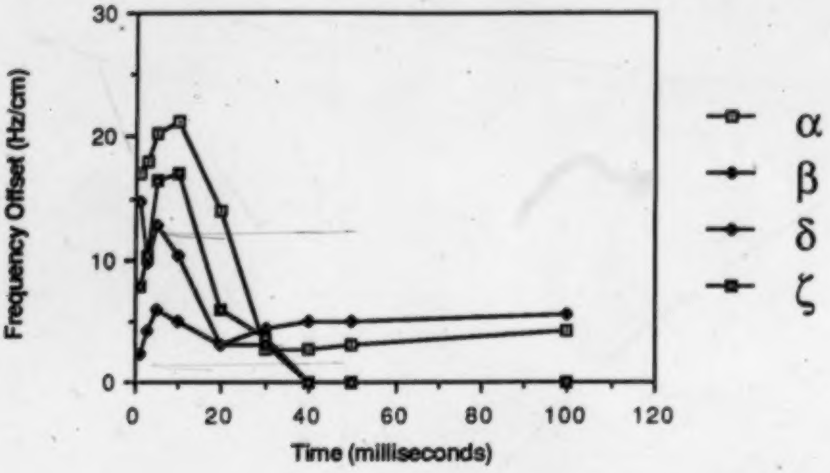


Figure 6.13 A plot, for each of the four gradient pulse sets, of the magnetic field gradient in Hz/cm between the two spheres over time. The time origin is at the end of the gradient pulse set

In summing up the three experiments, it is clear that the first experiment shows that the axial linearity of the z direction magentic field gradient is very good. Thus axial non-linearity is not a contibuting factor to the aforementioned poor image quality. The other two experiments do, however, show some serious problems. The radial dependence of the magnetic field gradient will be difficult to correct. Perhaps thin strips of metal could be used to " shim " this magnetic field gradient without disturbing the other magnetic field gradients and the homogeneity of the main magnetic field. Pulsed gradient induced eddy currents can be minimized by proper adjustment of the gradient pulse envelope characteristics. That is, proper adjustment of the rise and fall times of the pulse edges and perhaps also the shape of the pulse envelope.

The observed magnetic field gradients generated by eddy currents in the cryostat do not necessarily have a great effect on the slice selection process. This statement is supported by the fact that the magnetic field gradient generated by pulse set δ is clearly lower in magnitude over time than that generated by pulse set α (see Figures 6.12 and 6.13). Pulse set δ mimics the slice selection gradient pulse set that resulted in image artifacts. However, these eddy current-generated magnetic fields, and similar fields generated by the switching of the x and y direction magnetic field gradients, can cause errors in phase images by producing unwanted phase encoding during the imaging experiment.

The demonstrated radial asymmetry of the z direction magnetic field gradient, on the other hand, provides a strong argument against using a slice selection gradient pulse followed immediately by a slice refocussing gradient pulse in the opposite direction. As illustrated in Figure 6.11, there will be improper refocussing of the spins. Further work will be required to quantify, explain and compensate for this refocussing problem.

Further imaging pulse sequence tests showed that a return to a slice selection sequence with the slice refocussing pulse after the 180 degree RF pulse removed the previous image artifacts.

6.3 Flow Velocity Measurements

Having to return to the " old " slice defining gradient pulse sequence (the slice refocussing gradient pulse after the 180 degree RF pulse) meant that some other method had to be found to compensate for the large phase angles encoded for flow by the slice selection gradients. At this time a closer look was taken at the theory behind phase encoding due to constant velocity motion (section 3.2). This resulted in the inclusion of a compensating " balanced " gradient pulse set between the 90 and 180 degree RF pulses. This balanced gradient pulse set compensates for or removes much of the phase encoding of the moving magnetization incurred by the slice selection gradient pulse sequence (refer to equations 3.13 and 3.14). The final form of the flow measurement pulse program is shown in Figure 6.14.

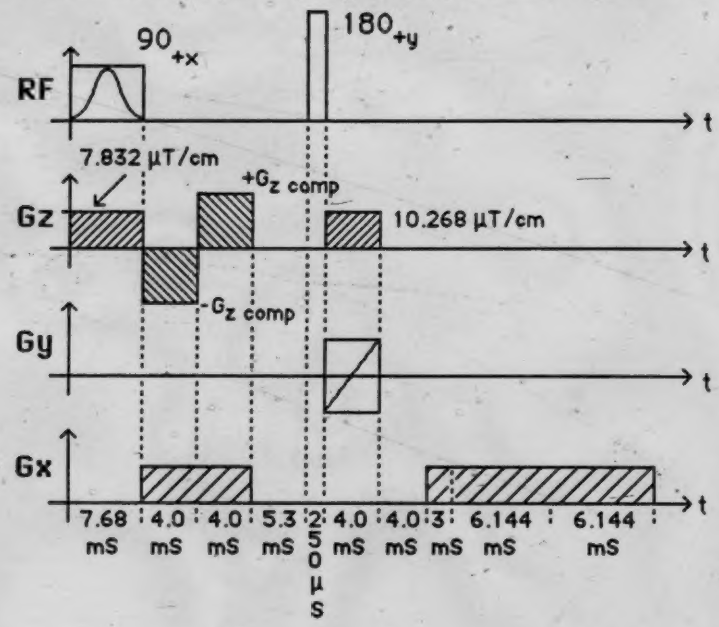

90 +x
180 +y
RF
t
7.832 µT/cm
+Gz comp
Gz
10.268 µT/cm
t
-Gz comp
Gy
t
Gx
t
7.68 mS
4.0 mS
4.0 mS
5.3 mS
250 µS
4.0 mS
4.0 mS
3 mS
6.144 mS
6.144 mS

Figure 6.14 The final flow measurement pulse program. The amplitudes of the x and y gradient pulses are dependent on the " magnification " factor desired. The value of $G_{z\,comp}$ depends on the range of flow rates to be measured.

Referring to the end of section 3.2, the phase of magnetization moving along the z axis should be given by

$$\phi = \gamma G_z v_z \left(\frac{59}{50} t_p^2 + t_p \Delta t_z \right) - \gamma G_{z\,comp} v_z t_p^2 \qquad 6.2)$$

where, from Figure 6.14, t_p is 4 milliseconds and G_z is $10.268 \cdot 10^{-6}$ Tesla cm^{-1}.

To check the validity of this " flow phase compensation " scheme, an experiment was performed as follows. The U-tube phantom was set up and the spectrometer prepared for imaging using the flow measurement pulse program with the compensating balanced gradient set installed. The spin echo was centred in the sampling interval using IMFLTR. A constant flow was established through the U-tube phantom. Image data was acquired for several IOD values of $G_{z\ comp}$ (Figure 6.14). These images were reconstructed and the phase of the signal from the flowing water measured. The measured values were normalized to the flow rate during the experiment (see Table 6.2) and are plotted in Figure 6.15.

Image	Gradient (IOD)	Flow rate (cm/s)	Measured phase (degrees)	Phase divided by flow rate (degrees s cm^{-1})
A	0	5.1	-55.1	-10.8
B	50	5.1	-45.9	-9.0
C	100	5.1	-43.4	-8.5
D	300	5.3	-8.5	-1.6
E	400	6.2	3.8	0.7
F	500	5.1	19.5	3.8
G	600	5.0	36.8	7.4

Table 6.2 Results of the slice selection gradient flow phase compensation test.

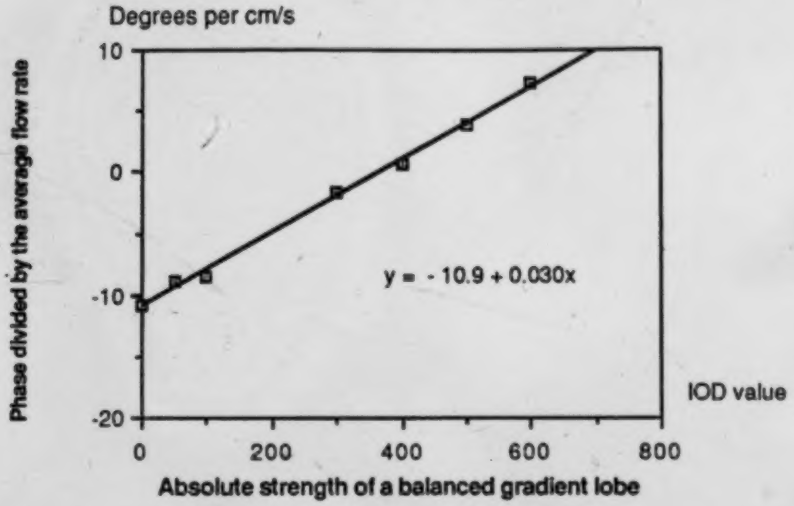


Figure 6.15 The effect of different flow compensating gradient pulse amplitudes (in IOD values of one lobe of the gradient set, the other is the same magnitude and in the opposite direction) on the encoded phase per cm s^{-1}.

The slope of the least squares fitted line is 0.030 ± 0.0011 degrees s cm^{-1} IOD^{-1}. The intercept of -10.9 ± 0.4 degrees s cm^{-1} is in good agreement with that which should be encoded at zero flow compensating gradient. Using equation 3.10) and values from Figure 6.14, this would be

$$\frac{\phi}{v_z} = \gamma G_z \frac{1}{2}\left(\frac{59}{25} t_p^2 + 2 t_p \Delta t_z\right)$$

$$= \gamma 10.26 \cdot 10^{-6} \frac{1}{2}\left(\frac{59}{25} 16 \cdot 10^{-6} + 108.43 \cdot 10^{-6}\right)$$

$$= 0.20 \text{ radians s cm}^{-1} \text{ or } 11.49 \text{ degrees s cm}^{-1}$$

The coefficient of determination is 0.9967. However, for IOD values greater than 500 distortions in the phase and modulus images began to appear.

This is not a problem since any flow rate of physiological interest can be measure using compensating gradients in the range of $0 \leq \text{IOD} \leq 400$.

Using an IOD value of 400 for the balanced gradient pulse set and the values shown in Figure 6.14 for the slice defining gradient pulses, trials were conducted to determine the precision of the phase measurement for flowing and non-flowing liquid. The flow was set to a constant rate and a set of six images were made. Following reconstruction, the phase of the water flowing through the inlet tube was measured for each image. The average phase of the water surrounding the inlet tube was measured as well as the phase of the flowing water in the inlet tube. The former value was subtracted from the latter and the result called the phase of the flowing water. Typical results are tabulated in Table 6.3.

Image	Flow rate (cm/s)	Measured phase (degrees)	Phase for a flow rate normalized to 9.0 cm/s (degrees)	Average phase of H_2O around the inlet tube (degrees)
A	10.8	-6.5	-5.4	5.5
B	9.7	-5.2	-4.8	5.3
C	9.0	-5.1	-5.1	6.5
D	8.5	-3.6	-3.8	9.5
E	8.1	-1.9	-2.1	8.3
F	7.6	-1.5	-1.8	7.3
	$v_{avg} = 9.0$ ± 1.2 cm/s		$\phi_{flow} = -3.8$ ± 1.6 degrees	$\phi_{stat} = 7.1$ ± 1.6 degrees

Table 6.3 Results from test of flow measurement precision.

The precision of the phase measurement for both flowing and non-flowing water is within ± 2 degrees.

At first glance the precision does not appear to be satisfactory since the measured phase values for flow are so low (i.e., a variation of 2 degrees in 5 degrees gives a 40% error). However, it must be remembered that this flow phase compensating balanced gradient set was chosen to allow encoding of flow rates up to 150 ml s^{-1}. Blood, having a higher viscosity than water, will maintain laminar flow at much higher flow rates (McDonald, 1974). At flow rates of 120 cm s^{-1} the phase encoded would be about 50 degrees. A variation of 2 degrees at that rate gives and error of 4%. The fact that the variation in the measurement of the phase for static water is about the same as the variation in the phase measurement for flowing water suggests the error in the flow measurement (section 5.1) is not a significant factor in the overall phase measurement error.

It is interesting to note the almost linear decrease in the measured phase for flowing water from image to image. The data is presented in chronological order with each image in the sequence requiring about 6 minutes to acquire. This decrease will be examined towards the end of the chapter.

The next step, after the precision of the phase measurement was found, was to find the accuracy of the flow measurements. A compensating balanced gradient set of IOD 400 was used. Images for several different flow rates were generated, reconstructed and the phase of the water flowing in the inlet tube measured. The measurement results and the least squares linearly fitted data are presented in Table 6.4 and plotted in Figure 6.16. A profile across the inlet tube (Figure 6.17) shows the laminar flow distribution. Confidence intervals of 95% were calculated using a Student's t distribution for the straight line parameters. These parameters

Flow rate (cm/s) calculated	Measured phase (degrees)	Phase calculated for fitted line (degrees)	Measured phase minus the calculated phase (degrees)
0	-2.3	-1.7	-0.6
3.9	-19.2	-20.3	1.1
4.7	-24.5	-24.1	-0.4
4.8	-25.0	-24.6	-0.4
5.9	-29.4	-29.8	0.4
6.2	-30.2	-31.3	1.1
6.7	-30.8	-33.7	2.7
6.7	-32.1	-33.7	1.6
7.2	-32.9	-36.1	3.0
9.6	-54.3	-47.5	-6.8
9.6	-55.3	-47.5	-7.8
13.0	-57.6	-63.8	6.2

Table 6.4 Results of flow measurement accuracy test.

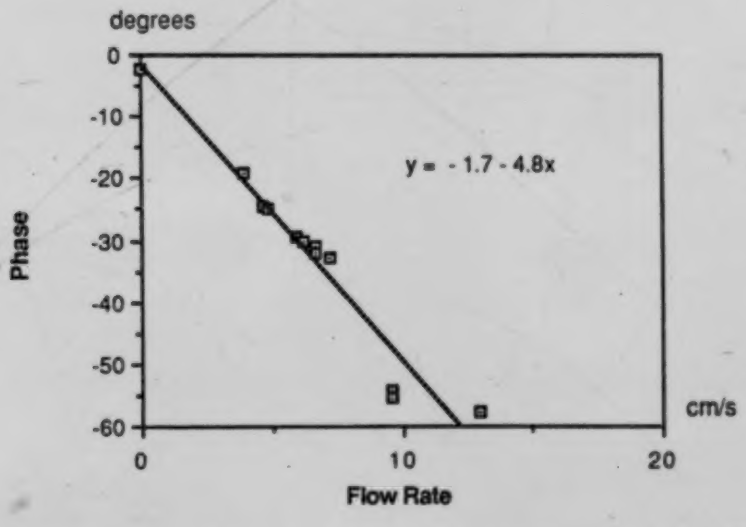


Figure 6.16 The relationship between flow velocity and the measured phase for a compensating gradient pulse amplitude of IOD = 400 in the pulse program shown in Figure 6.14.

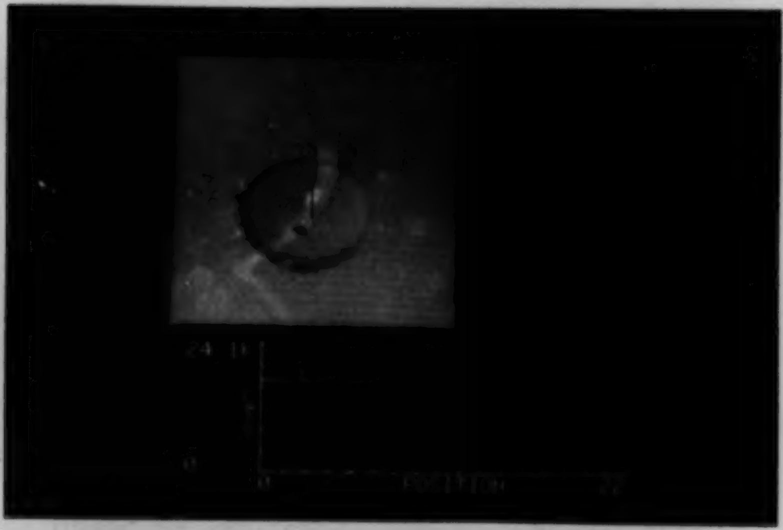

0
0
POSITION

Figure 6.17 A profile across the inlet tube showing the laminar flow velocity distribution.

and their associated intervals are: a slope of -4.8 ± 0.4 degrees s cm^{-1} and an intercept of -1.7 ± 2.6 degrees. The coefficient of determination is 0.941 and the variance is 16.85. The expected value for the intercept (zero degrees phase difference for a flow of zero cm s^{-1}) falls within the interval about the calculated intercept. However, the confidence interval about the value of the slope is quite large at ± 7%. Examination of the measured phase points and the fitted data points in Table 6.4 reveals that the differences increase steadily with flow rate. This is most certainly the factor determining the uncertainty in the slope. This increase in the differences could be due to the onset of turbulent flow in the inlet tube. In section 5.1 it was said that flow rates in all experiments were almost always kept below 10 cm s^{-1}, well below the Reynolds number of 1600. However, due to the nature of the phantom construction and its operation, this condition may not have been sufficient. For example, no measurements were made for the outlet tube. This is because there was a sharp bend at the base of the U-tube which caused turbulent flow patterns (non-laminar) visible in the phase images. During the course of an imaging session air bubbles would form inside the tubes, gradually restricting the flow and in some cases, when they formed near the imaging slice location, causing turbulent flow. At the juncture of the glass and flexible tubes there was a sharp glass edge (the flexible tubing was pushed over the glass tube) at which vortices could form and be carried downstream into the imaging slice. Naturally, these problems would be aggravated at higher flow rates.

A final possibility is the gradual decrease of drift in the measured phase for a given flow rate as witnessed earlier. This problem is discussed more fully at the end of this section.

The question of precision and accuracy should be placed into a physiological perspective. Ultrasonic Doppler Flow meters have been used extensively in the past few years to measure blood flow. Uematsu et al. (1983) report an error in the measurement of blood flow of about 5% using an ultrasonic volume flow meter. Their paper then goes on to describe the variability of common carotid blood flow in normal volunteers on an " intrasession " (within one sitting) and an " intersession " (week to week) basis. They found that the standard deviation of a repeated intrasession measurement was 3.8%. This means that repeated measurements of the flow through the carotid could give readings within plus or minus 7.6% (two standard deviations). The standard deviation of the intersession measurement was 10.6%. This may seem large, but, as Uematsu et al. go on to state, it is within the normal flow rate of 6 ml s^{-1} to 11 ml s^{-1}. Recalling that the error in the slope of the flow measurement curve in Figure 6.16 is ± 7%, a simple calculation will indicate it's seriousness. Assuming the worst possible case, that of a 14% error in the slope, consider a flow of 8.0 ml s^{-1} through the carotid of a human aged 21 to 40 with a mean diameter of 7.2 mm (Uematsu et al., 1983). The average flow velocity would be 19.65 cm s^{-1} (this analysis assumes that some form of cardiac synchronization has been implemented). An error of 14% in the NMR imaging flow measurement of this flow would indicate a " flow rate " from 16.9 cm s^{-1} to 22.4 cm s^{-1} or a " flow volume " of 6.88 ml s^{-1} to 9.12 ml s^{-1}. These values are still within the reported range of normal flow volumes for that age group of 7.99 ± 1.5 ml s^{-1}, albeit not by much.

At this point a short investigation of the effect of turbulent flow on an image was made. Based on the work of Wood and Henkelman (1985), a model was developed to explain the appearance of artifacts once flow becomes turbulent or pulsatile (refer to section 3.3 and Appendix B). A simple experiment using the U-tube flow phantom was performed. Flow images were generated at average flow rates in the range of 3 cm s^{-1} to 6 cm s^{-1} and 9.71 cm s^{-1} to 27 cm s^{-1}. Figure 6.18a shows a phase image with flow at 3.8 cm s^{-1} (Re = 300) which is definitely constant and laminar. In Figure 6.18b the flow rate had been increased to 26.1 cm s^{-1} (Re = 2000), the supposed onset of turbulent flow. The artifacts are visible in the phase encoding direction in a band the width of the tubes as predicted by section 3.3. The artifacts on either side of the outlet tube in Figure 6.18a are probably due to the previously mentioned bend at the base of the U-tube. The fact that they are equally spaced and well defined suggests that this bend produces a simple rhythmic pulsation in the flow in the outlet tube.

The final question addressed in this project was that of reproducibility. This would determine the overall flow measurement procedure. If the results from phantom tests were constant from week to week or month to month, then the flow measurement technique would need to be calibrated a few times a year. However, if the slopes of the calibration curves (slope in degrees s cm^{-1}) varied from day to day or week to week, then the flow measurement experimental procedure would have to be modified to include a calibration step.

This question was approached in two stages. The first stage looked at the variations in the slope over a short period of time(hours). The second stage examined the change in the slope over a period of days.

The first stage of experiments proceeded as follows. Two separate flow measurement experiments were run in one day. In each experiment a constant flow rate of about 5 cm s^{-1} was established through the U-tube phantom. Six images were generated in each experiment at this flow rate,

Figure 6.18 a) A phase image showing slow (3.8 cm/s) laminar flow. The artifacts on either side of the upper, outflow tube, are due to turbulence generated by the sharp bend at the base of the U-tube.

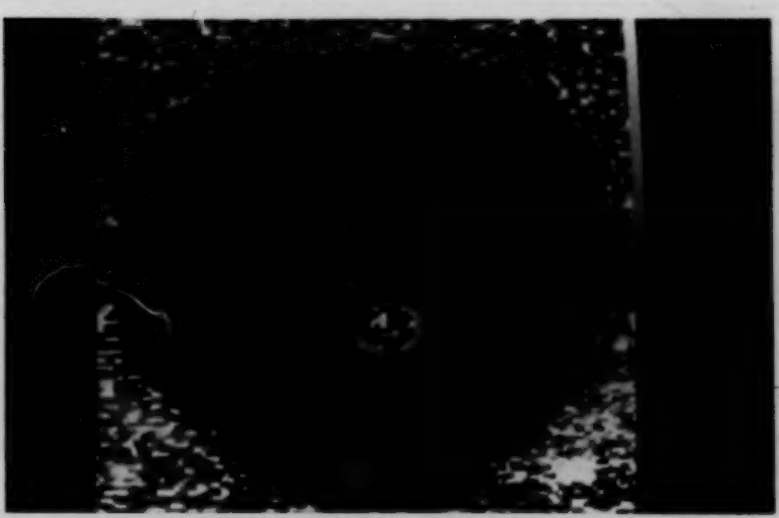
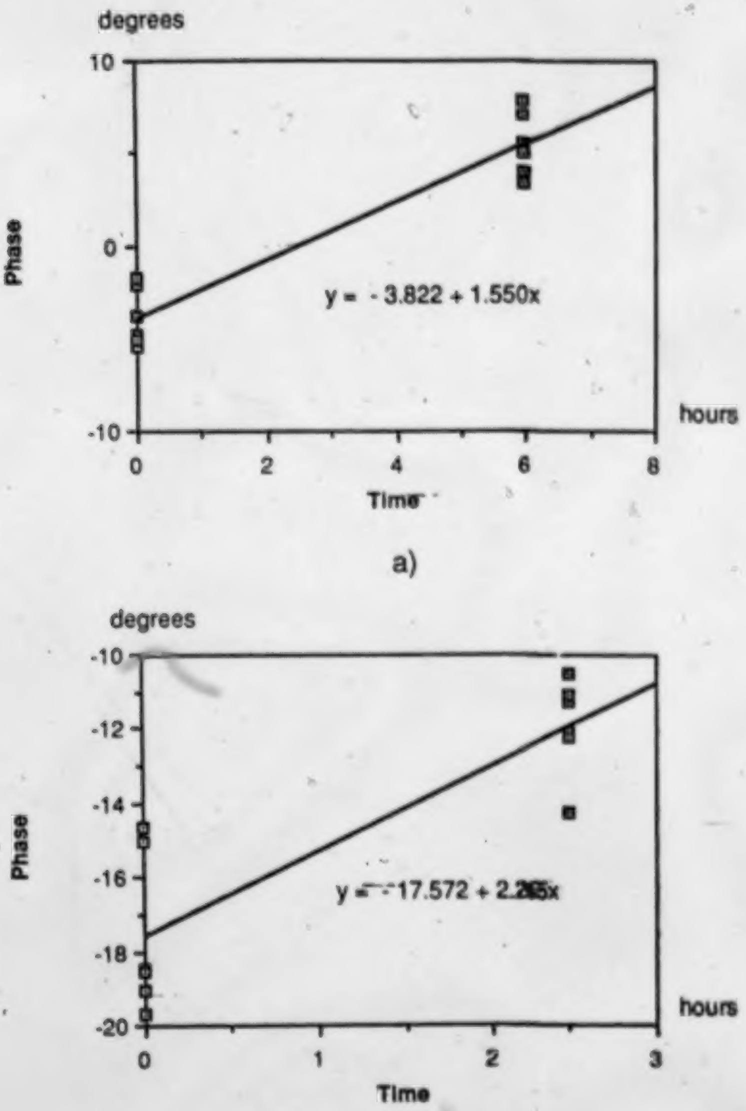


Figure 6.18 b) A phase image of the same phantom showing a high flow rate (26.1 cm/s) and turbulence in both tubes. The artifacts along the phase encoding direction are due to the variability of the flow velocity components during the imaging experiment.

each image requiring about 7 minutes to acquire. The total time for one, six image experiment, was about one hour. The two six-image experiments were separated in time (a few hours) during which the spectrometer was left on. This was repeated on two separate days using a different $G_{z\ comp}$ balanced gradient set each day. The resulting image data were reconstructed and the phase of the water flowing through the inlet tube measured.

The measured phases were normalized to the average flow rate of the day's experiment. The results have been plotted in Figure 6.19. It is clear in both plots, i.e., for both days, that there is a drift over time of the measured phase for a given flow rate.

The second stage of the test involved the measurement of the phase encoded by a given flow rate over a period of days. Since this last stage has taken place near the end of the project, it was decided to use values measured over the previous month (September, 1986). A set of phase values were chosen such that the flow rates at which they were measured averaged to 4.8 ± 0.5 cm s^{-1}. The phase values were normalized to this average flow rate. These measurements are taken from different days, with several points being measured in each day. The result of this is shown in Figure 6.20.



a)

b)

Figure 6.19 a) The change in the measured phase on day zero (average flow rate = 4.5 cm s^{-1}) over time. The flow compensating gradient pulse amplitude was IOD = 400.

b) The change in the measured phase on day five (average flow rate = 5.6 cm s^{-1}) over time. The flow compensating gradient pulse amplitude was IOD = 200.

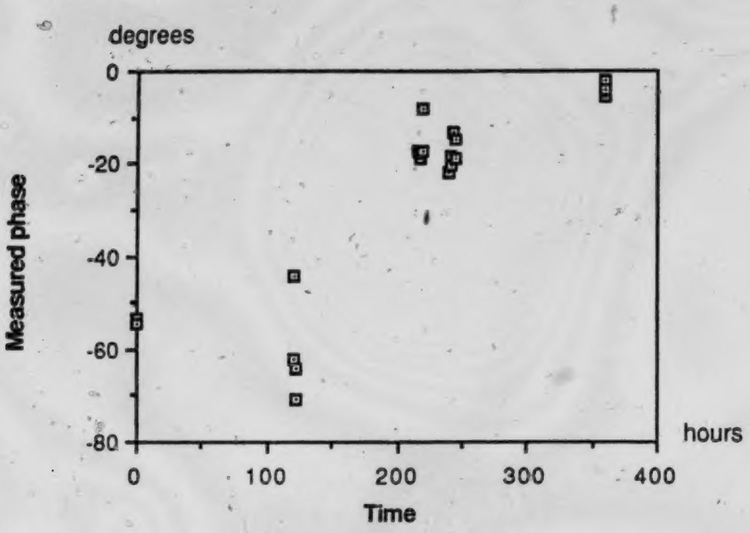


Figure 6.20 The change in the measured phase (average flow rate is 4.8 cm s^{-1}) over several days. The flow compensating gradient pulse set amplitude was IOD = 400.

As evidenced by Figure 6.20, the phase measured for a more or less constant flow rate changes by a large amount over a relatively short period of time. Yet, over this period of time, flow calibration curves, that is, straight line fits of phase data for different flow rates, still yielded coefficients of determination of 0.94 and better within a given day. This indicated that the performance of the flow measurement system over a short period of time is linear.

The fluctuation of the phase encoded by a given flow rate over a period of time as shown in Figure 6.20 implies that, while the technique is relatively stable over a short time (hours), it is not stable over a longer time (day to day). The cause (or causes) behind this fluctuation have not been found, but some variables have been eliminated. Variations in the measured phase due to phantom set up procedures and spectrometer initialization (i.e., preparation of the spectrometer for imaging) have been demonstrated to be not greater than the " normal " phase measurement error of 2%.

In light of this " drift " in the measured phase for a particular flow rate, a second experiment measuring the effect of different balanced gradient sets between the 90 and 180 degrees RF pulses was performed. Again, a constant flow rate was established and the spectrometer prepared to image. Images were generated for several different balanced gradient IOD values and the phase of the flowing water measured in each. These values were normalized by the average flow rate (5.1 cm s^{-1}) and then plotted against the balanced gradient strength (Figure 6.21). The slope of the curve was found to be 0.030 ± 0.0033 degrees s cm^{-1} IOD^{-1} and the intercept was -15.7 ± 1.1 degrees s cm^{-1}. The coefficient of determination was 0.973. The slopes of the curves in Figures 6.15 and 6.21 can be seen to be statistically the same, but the intercepts are markedly different. This implies that the effective strength of the z direction magnetic field gradient pulses (slice selection and flow compensation) changed over the seven days separating the two experiments. This could be a physical change in the strength of the gradient magnetic field along z or some other change in the magnetic system that affects how the spins respond to the z direction magnetic field gradient.

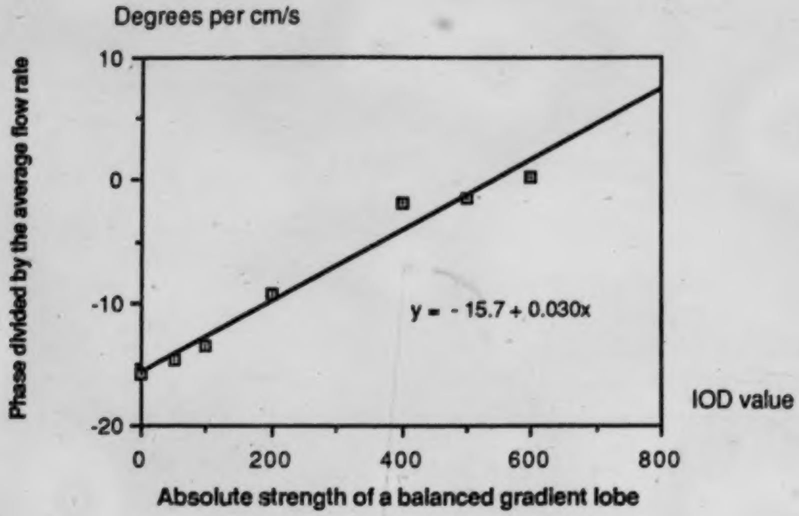


Figure 6.21 Results of the second experiment investigating the effect of different flow compensating gradient pulse amplitudes (in IOD values) on the phase encoded. Notice the " drift " downwards of the intercept value in comparison with Figure 6.15.

In any case it is certain that more experiments must be performed to establish the trend of this " drift ". Since phase is cyclic, this drift cannot be linear. These experiments must necessarily take place at weekly intervals over a period of months, which precludes any further work as far as this project is concerned.

6.4 A Suggested Experimental Method for Flow Velocity Measurement and Further Experiments

The fact that the phase measured for a given flow rate changes over time suggests that the technique must be calibrated before or during a set of measurements on an animal or a human. The experiment would begin with the estimation of the peak flow velocity likely to be measured. Then a z direction gradient set (slice selection and flow compensation gradient set) that would properly encode the anticipated peak flow rate would be chosen. Since the drift in the phase measured for a given velocity can be as great as 0.4 degrees per cm s^{-1} per hour, a method of flow rate calibration *during* the experiment would have to exist. One possibility is the inclusion of a straight tube in the bore of the magnet with the subject. Very constant flow would be maintained through the tube during the flow measurement experiment. Since, over a short time, the relationship between phase and flow velocity is very linear, measurement of the phase of the flow in the tube divided by the flow velocity should give the slope of the calibration curve from which flow velocity values could be found based on phase measurements from the image of the subject. It is also crucial that the spin echo be centred in its sampling window for each new subject. This, as mentioned earlier, is easily done using IMFLTR.

Further investigations into the nature of the " phase drift " should concentrate on the change in the effect of the z direction gradient pulses over time. In addition to the experiment that was repeated at the end of section 6.3, a second, more direct experiment could be pursued. It would involve the measurement of the slice encoding gradient strength over a period of weeks. Two small spheres of water would be fixed a set distance either side of the centre of the magnet along the z axis. A z direction gradient would be established (same strength, as defined in software, each week). The spheres would be irradiated by a short 90 degree RF pulse and the resulting FID acquired. Analysis of the frequency spectrum of the FID would give the separation of the two spheres in Hz. This separation could be monitored from week to week to give an accurate measure of the change in the effective gradient strength. This, in conjunction with the previously mentioned experiment, should provide some clues as to the cause of the measured phase drift.

7. Conclusions

This thesis has endeavored to present the work done on fluid flow measurement over the past two years at the medium bore animal NMR unit of the University of Alberta. These two years represent a stepwise theoretical and experimental progression in the understanding and investigation of flow measurement using NMR imaging. This progression is reflected in the organization of the thesis and, specifically, in the structure of Chapter 6.

In pursuit of a technique that would allow the measurement of physiologically significant blood flow rates, many insights, some new, have been made concerning NMR imaging. Early work involving analysis of the phase errors across a spin echo generated image of stationary water revealed the dependence of the phase on the temporal position of the spin echo. This was followed by an investigation of the slice defining sequence that led, eventually, to the discovery of the radial dependence of the slice defining magnetic field gradient. This was not expected and resulted in a change in the flow measurement technique implementation. Other " side " investigations, such as magnetic susceptibility visualization experiments and gradient induced eddy current measurements have yielded results not only useful in this thesis, but useful to the animal unit as a whole.

The choice of a flow measurement technique relying on the change in the phase of the NMR signal from magnetization that moves during the imaging experiment required that a mathematical analysis be made of the phase of the NMR signal in such an experiment. This analysis provided a clear understanding of how the phase would change for constant velocity motion along any of the three imaging axes. It lead to the implementation of flow compensating gradient pulses in the pulse program and was the basis for a new model describing image artifacts due to periodic and non-constant velocity motion.

Considering the aforementioned mathematical analysis of image phase, there exists the possibility of signal phase manipulation by including " special " gradient pulses in imaging pulse programs. In principle, any factor that adversely affects the signal phase, such as motion due to regular breathing, could be eliminated or uniquely specified as desired.

These insights have led to an increased understanding of the NMR imaging experiment and should provide useful information for future work in flow measurement and other imaging techniques.

The flow measurement technique developed in this project has been demonstrated to function over a short period of time. Changes due to flow in the phase of a region in an image can be measured with good precision. However, there is a serious problem with flow measurement variability over a long term. The inclusion of a flow phantom with the subject in the bore of the magnet is an interim solution to this problem. It may not always be possible or desirable to do this. The investigation of the variability of the flow phase measurement is but another stage in the project, albeit a stage that follows the work in this thesis. There is also room for improvement in the calibration method (by calibration it is meant the measurement of the phase for accurately known flow rates to establish the slope and the intercept of the phase versus flow rate curve). It would be desirable to minimize the variation in the flow through a phantom during an experiment. This would be especially important during flow measurement imaging of a living subject. The gravity pressure flow setup would have to be replaced by a constant volume, non-pulsatile flow pump. In addition, it would make sense to replace tap water by something with viscosity similar to blood so that laminar flow could be maintained at higher flow rates.

A consideration for further *in vivo* work is the effective minimum vessel diameter for which the technique will yield accurate results. Throughout the project a phantom with tubes of large diameter has been used. Such large regular cross sections would not be present in the image of a living subject. Vessel cross sections in NMR are not usually circular, thus some method will have to be developed to define an irregularly shaped region of interest and then measure the phase within it. This is similar to the problem in bone density Computed Tomography of defining the extent of the cortical bone in a cross sectional image of a limb.

Even in its current state, the flow measurement technique implemented in this thesis could provide useful *in vivo* blood flow information. This technique offers advantages over other techniques, such as Ultrasonic Doppler methods, in its ability to see deep within the body and " past " bone structure. However, if measurements are to be made on arteries or in the region of the heart, cardiac synchronization must be included in the flow measurement imaging experiment. It is interesting to note that, although medically related NMR publications are full of papers describing phantom flow measurements and preliminary *in vivo* results, there are no papers in which extensive *in vivo* blood flow measurement studies have been made. This suggests that there is either insufficient interest in the technique or that other researchers are encountering difficulties similar to the ones mentioned here. Nevertheless, the demonstrated precision and short term accuracy of the phase encoding technique does provide incentive for further development work.

References

Arnold DW, Burkhart LE. Spin-echo NMR response from a flowing sample. J Appl Phys **1965**; 36:870-871

Axel L. Blood flow effects in magnetic resonance imaging. AJR **1984**; 143:1157-1166

Bendall MR. Private communication. **1986**

Bloch F, Hansen WW, Packard M. Nuclear induction. Phys Rev **1946a**; 69:127

Bloch F, Hansen WW, Packard M. The Nuclear induction experiment. Phys Rev **1946b**; 70:474-485

Bloch F. Nuclear induction. Phys Rev. **1946c**; 70:460-473

Bottomley PA, Andrew ER. RF magnetic field penetration, phase shift and power dissipation in biological tissue: implications for NMR imaging. Phys Med Biol **1978**; 23:630-643

Bottomley PA, Edelstein WA. Power deposition in whole-body NMR imaging. Med Phys **1981**; 8:510-512

Bottomley PA, Redington RW, Edelstein WA, Schenck JF. Estimating radiofrequency power deposition in body NMR imaging. Mag Res Med **1985**; 2:336-349

Bradley WG, Waluch V, Ka-Siu L, Fernauder EJ, Spalter C. The appearence of rapidly flowing blood on magnetic resonance images. AJR **1984**; 143:1157-1174

Brandes R, Kearns DR. Generation of tailored radiofrequency pulses by a simple audiofrequency filter method. II. Analysis. J Mag Res **1986**; 67:14-27

Bryant DJ, Payne JA, Firmin DN, Longmore DB. Measurement of flow with NMR imaging using a gradient pulse and phase difference technique. J Comput Assist Tomogr **1984**; 8:588-593

Carr HY, Purcell EM. Effects of diffusion on free precession in nuclear magnetic resonance experiments. Phys Rev **1954**; 94:630-638

Constantinesco A, Mallet JJ, Bonmartin A. NMR phase modulation method for flow analysis. Abstract, 1st Congress of the European Society of Nuclear Magnetic Resonance in Medicine, Geneva, October 5,6, 1984

Crooks LE, Hoenninger JC, Arakawa M. Pulse sequences for NMR imaging using multidimensional reconstruction techniques. In: Partain CL, ed. Nuclear magnetic resonacne and correlative imaging modalities. New York: Society of Nuclear Medicine, **1984**:69-74

Damadian R. Tumour detection by nuclear magnetic resonance. Science **1971**; 171:1151-1153

Deimling M, Mueller E, Lenz G, Barth K, Fritschy P, Seiderer M, Reinhardt ER. Description of flow phenomena in magnetic resonance imaging. Diag Imag clin Med **1986**; 55:37-51

Feinberg DA, Crooks LE, Sheldon P, Hoenninger J, Watts J, Arakawa M. Magnetic resonance imaging the velocity components of fluid flow. Mag Res in Med **1985**; 2:555-566

Garroway AN. Velocity measurements in flowing fluids by NMR. J Phys D **1974**; 7:L159-L163

Glover GH, Hayes CE, Pelc NJ, Edelstein WA, Mueller DM, Hart HR, Hardy CJ, O'Donnel M, Barber WD. Comparison of linear and circular polarization for magnetic resonance imaging. J Mag Res **1985**; 64:255-270

Grover T, Singer JR. NMR spin-echo flow measurements. J Appl Phys **1971**; 42:938-940

Hahn EL. Spin echoes. Phys Rev **1950**; 80:580-594

Hahn EL. Detection of sea-water motion by nuclear magnetic precession. J Geophys Res **1960**; 65:776-777

Hayward RJ, Packer KJ, Tomlinson DJ. Pulse field-gradient spin echo. N.M.R. studies of flow in fluids. Molecular Phys **1972**; 23:1083-1102

Hinshaw WS, Bottomley PA, Holland GN. Radiographic thin-section image of the human wrist by nuclear magnetic resonance. Nature (London) **1977**; 270:722-723

Kumar A, Welti D, Ernst RR. NMR fourier zeugmatography. J Mag Res **1975**; 18:69-83

Lauterbur PG. Image formation by induced local interactions: examples employing nuclear magnetic resonance. Nature **1973**; 242:190-191

Locher PR. Computer simulation of selective excitation in n.m.r. imaging. Phil Trans R Soc Lond **1980**; 289:537-542

Mansfield P, Grannell PK. NMR ' diffraction ' in solids? J Phys C: Solid State Phys **1973**; 6:L422-L426

Mansfield P, Maudsley AA, Morris PG, Pykett IL. Selective pulses in NMR imaging: A reply to criticism. J Mag Res **1979**; 33:261-274

Meiboom S, Gill D. Modified spin echo method for measuring nuclear relaxation time. Rev Sci Instr **1958**; 29:688

Mueller E, Deimling M, Reinhardt ER. Quantification of pulsatile flow in MRI by an analysis of T2 changes in ECG-gated multiecho experiments. Mag Res in Med **1986**; 3:331-335

Norris DG. Phase errors in NMR images. " Book of Abstracts, Fourth Annual Meeting of the Society of Magnetic Resonance in Medicine, London, United Kingdom, August 19-23, 1985, " pp. 1037-1038 (abstract)

O'Donnel M. NMR blood flow imaging using multi-echo phase contrast sequences. Med Phys **1985**; 12:59-64

Packer KJ. The study of slow coherent molecular motion by pulsed nuclear magntic resonance. Molecular Phys **1969**; 17:355-368

Purcell EM, Torrey HC, Pound RV. Resonance absorption by nuclear magnetic moments in a solid. Phys Rev **1946**; 69:37

Singer JR. Blood flow rates by nuclear magnetic resonance measurements. Science **1959**; 130:1652-1653

Singer JR, Crooks LE. Nuclear magnetic resonance blood flow measurements in the human brain. Science **1983**; 221:654-656

Stejskal EO. Use of echoes in a pulsed magnetic-field gradient to study anisotropic, restricted diffusion and flow. J Chem Phys **1965**; 43:3597-3603

Summers RM, Axel L, Solomon I. A computer simulation of nuclear magnetic resonance imaging. Mag Res in Med **1986**; 3:363-376

Suryan G. Nuclear resonance in flowing liquids. Proc Indian Acad Sci [A] **1951**; 33:107-111

Sutherland RJ, Hutchison JMS. Three-dimensional NMR imaging using selective excitation. J Phys E Sci Instrum **1978**; 11:79-83

Torrey HC. Bloch equations with diffusion terms. Phys Rev **1956**; 104:563-565

Uematsu S, Yang A, Preziosi TJ, Kouba R, Toung TJK. Measurement of carotid blood flow in man and its clinical application. Stroke **1983**; 14:256-265

Van Dijk P. Direct cardiac NMR imaging of heart wall and blood flow velocity. J Comput Assist Tomogr **1984**; 8:429-436

Wedeen VJ, Meuli RA, Edelman RR, Geller SC, Frank LP, Bradly TJ, Rosen BR. Projective imaging of pulsatile flow with magnetic resonance. Science **1985**; 230:946-948

Wood ML, Henkelman RM. MR image artifacts from periodic motion. Med Phys **1985**; 12(2):143-151

Zimmerman JR, Brittin WE. Nuclear magnetic resonance studies in multiple phase systems: lifetime of a water molecule in an adsorbing phase on silica gel. J Phys Chem **1957**; 61:1328-1333

Bibliography

Abragam A. Principles of Nuclear Magnetism. (Clarendon Press, Oxford, **1961**)

Abramowitz M, Stegun IA, ed. Handbook of Mathematical Functions. National Bureau of Standard Applied Mathematics Series 55. (National Bureau of Standards { U.S. }, Washington, D.C., **1964**)

Burden RL, Faires JD, Reynolds AC. Numerical Analysis. (Prindle, Weber and Schmidt, Boston, **1981**)

Farrar TC, Becker ED. Pulse and Fourier Transform NMR. Introduction to Theory and Methods. (Academic Press, New York, **1971**)

Harris RK. Nuclear Magnetic Resonance Spectroscopy. (Pitman Books Ltd., London, **1983**)

McDonald DA. Blood Flow in Arteries. (Edward Arnold, London, **1974**)

Mansfield P, Morris PG. NMR Imaging in Biomedicine. (Academic Press, London, **1982**)

Rouse H. Elementary Mechanics of Fluids. (Dover, New York, **1978**)

Slichter CP. Principles of Magnetic Resonance. (Harper and Row, New York, **1963**)

Appendix A

In this appendix equation 3.9) is solved and cast into a form that gives the spatial frequency dependent terms. Equation 3.9) is

$$\theta = -\gamma\int_{t_p}^{2t_p} (G_x x + G_x v_x \tau + G_y y + G_y v_y \tau)\, d\tau - \gamma\int_{0}^{t_p} G_z z\, d\tau - \gamma\int_{\frac{3}{5}t_p}^{t_p} G_z v_z \tau\, d\tau$$

$$+ \gamma\int_{2t_p + \Delta t}^{2t_p + \Delta t + t} G_x (x + v_x \tau)\, d\tau + \gamma\int_{t_p + \Delta t_z}^{t_p + \Delta t_z + t_p} G_z (z + v_z \tau)\, d\tau \qquad \text{A.1)}$$

Breaking this equation into three components yields

$$\phi = \phi_x + \phi_y + \phi_z \qquad \text{A.2)}$$

Solving for ϕ_x gives

$$\phi_x = -\gamma G_x \left[x t_p + \frac{3 v_x}{2} t_p^2 \right]$$

$$+ \gamma G_x \left[x t + \frac{v_x}{2} \left\{ (2t_p + \Delta t + t)^2 - (2t_p + \Delta t)^2 \right\} \right]$$

$$= -\gamma G_x \left[x t_p + \frac{3 v_x}{2} t_p^2 \right] + \gamma G_x \left[x t + \frac{v_x}{2} \left\{ 4 t_p t + 2 \Delta t\, t + t^2 \right\} \right] \qquad \text{A.3)}$$

Defining k_x as $\gamma G_x(t - t_p)$ and rearranging terms gives

$$\phi_x = \gamma G_x x(t - t_p) + \gamma G_x v_x/2 \, [\, 4t_p t + 2\Delta t\, t + t^2 - 3t_p^2 \,]$$

$$= x k_x + \frac{\gamma^2 G_x^2 v_x}{2\gamma G_x} (t - t_p)^2 + \gamma G_x \frac{v_x}{2} (6\, t_p\, t + 2\Delta t\, t - 4t_p^2)$$

$$= x k_x + \frac{v_x k_x^2}{2\gamma G_x} + v_x k_x (3t_p + \Delta t) + v_x \gamma G_x (t_p^2 + \Delta t\, t_p)$$

A.4)

Likewise, using $k_y = -\gamma G_y t_p$

$$\phi_y = -\gamma G_y \left[y t_p + \frac{3v_y}{2} t_p^2 \right]$$

$$= y k_y + v_y k_y \frac{3}{2} t_p$$

A.5)

Finally, for the last term

$$\phi_z = -\gamma G_z \left[z t_p + \frac{v_z}{2} \frac{16}{25} t_p^2 \right]$$

$$+ \gamma G_z \left[z t_p + \frac{v_z}{2} \left\{ (2t_p + \Delta t_z)^2 - (t_p + \Delta t_z)^2 \right\} \right]$$

$$= \gamma G_z \frac{v_z}{2} \left[\frac{59}{25} t_p^2 + 2 t_p \Delta t_z \right]$$

A.6)

Summing these three terms yields the final expression for the phase of the signal

$$\phi = \phi_x + \phi_y + \phi_z$$

$$= xk_x + yk_y + v_x k_x \left(3t_p + \Delta t \right) + v_x \gamma G_x \left(t_p^2 + \Delta t \, t_p \right) + v_y k_y \frac{3}{2} t_p + \frac{v_x k_x^2}{2\gamma G_x} + \gamma G_z \frac{v_z}{2} \left(\frac{59}{25} t_p^2 + 2 t_p \Delta t_z \right)$$

A.7)

Appendix B

In this appendix the effect on an image by non-constant velocity motion is derived. The two dimenstional Fourier transform image technique is considered. The paper by Wood and Henkelman (1985) upon which the following analysis is based, considers artifacts in images that are a result of the periodic motion of breathing. Simple point source magnetization phantoms which oscillated along one of the three laboratory frame axes were used theoretically and experimentally. Their position was a sinusoidal function of time. Likewise, pulsatile flow will be modelled here as a point source magnetization, moving in a straight line, whose velocity varies sinusoidally with time. In this case, the equation describing the position of the magnetization in time along the z axis is

$$z(t) = z + v_z[\, 1 + \sin(\, 2\pi F_z t\,)\,] \qquad \text{B.1)}$$

F_z is the frequency of the velocity fluctuation. This model is not completely accurate, but it will suffice to illustrate the argument here.

It is assumed in this developement that the time over which the first gradient pulse along the slice or z direction (G_z pulse) affects the phase of the signal is two fifths the duration of the second, refocussing G_z pulse (see section 4.6.3 or equation 3.10). Recalling the mathematical development of the 2DFT experiment in section 2.10, but now including a term for phase encoded by movement along the z axis, the phase of the signal for the first echo is found to be a combination of equations B.1) and 2.56). That is,

$$\phi(t, G_y) = \gamma \left\{ \begin{array}{l} G_x x t - G_y y t_y \\ + G_z v_z \left[1 + \sin(2\pi F_z T + \theta_z) \right] \left[\frac{59}{50} t_p^2 + t_p \Delta t_z \right] \end{array} \right\}$$

B.2)

The first two terms eventually lead to the expression $M_o \delta(x-x_0)\delta(y-y_0)$ in the point spread function as defined in section 2.10. This is the magnitude of the signal at some (x,y). θ_z in the third term represents the offset phase at t=0. If it is assumed that the velocity is constant over the sampling period, $N_x \Delta t$, then the velocity expression is a function of the repetition time. That is, it varies from scan to scan. Therefore, $T = G_y/\Delta G \; T_R$, where T_R is the time between scans (Recall from section 2.10 that the phase encoding gradient begins at $-(N_y/2)\Delta G$. Therefore the time origin for this analysis is at the centre scan of the experiment). Dropping, for now, the x and y terms in equation B.2) and remembering that t_p and Δt_z are fixed in duration over the course of the imaging experiment, $\phi(t,G_y)$ can be modified to be

$$\phi'(t,G_y) = \psi[\, 1 + \sin(2\pi F_z T_R G_y/\Delta G + \theta_z) \,] \qquad \text{B.3)}$$

$$\psi = \gamma G_z v_z \left(\frac{59}{50} t_p^2 + t_p \Delta t_z \right)$$

Using the identity (Abramowitz and Stegun, 1964),

$$\exp(\, i\, k \sin\theta \,) = \sum_{m=-\infty}^{\infty} \exp(\, i\, m\theta \,)\, J_m(k)$$

where J_m is the mth order Bessel function of the first kind, equation B.3), the exponential term in equation 2.65) becomes

$$\exp\left[i\,\phi'(t,G_y) \right] = \exp(i\psi) \sum_{m=-\infty}^{\infty} \exp(im\theta_z)\, J_m(\psi)\, \delta\left(f_y - mF_z \frac{T_R}{\Delta G} \right)$$

Including the x and y terms previously left out gives

$$\begin{aligned} &\mathbb{F}(\mathbb{F}\{M_o \exp[i\phi(t,G_y)]\}) = \\ &\quad \left[M_0 \delta(f_x - \frac{\gamma}{2\pi} G_x x_0)\delta(f_y + \frac{\gamma}{2\pi} t_y y_0) \right] \\ &\quad\quad \otimes_{f_y} \left[\exp(i\psi) \sum_{m=-\infty}^{\infty} \exp(im\theta_z)\, \mathbb{F}\left[J_m(\psi) \right] \delta\left(f_y - mF_z \frac{T_R}{\Delta G} \right) \right] \end{aligned}$$

B.4)

In terms of the spatial coordinates of the transverse magnetization (as given in section 2.10)

$$\begin{aligned} P(x,y) &= K[M_o \delta(x - x_o)\delta(y - y_o)] \\ &\quad \otimes_y \left[\exp(i\psi) \sum_{m=-\infty}^{\infty} \exp(im\theta_z)\, \mathbb{F}\left[J_m(\psi) \right] \delta(y - my_G) \right] \end{aligned}$$

B.5)

$$y_G = 2\pi F_z \frac{T_R}{\gamma \Delta G t_y}$$

$$J_m(\psi) = \sum_{r=0}^{\infty} \frac{(-1)^r \left(\frac{\psi}{2}\right)^{m+2r}}{r!\,\Gamma(m+r+1)}$$

$$\Gamma(n) = \int_0^{\infty} x^{n-1} e^{-x}\, dx$$

In equation B.5), the terms exp(iψ), exp(imθ_z) and $\mathbb{F}[J_m(\psi)]$ govern the magnitude and phase of the artifact. The term $\delta(y - my_G)$ determines the position of the artifact in the phase encoding direction. The spacing, y_G, is a function of the frequency, F_z, of the fluctuating velocity. In this simple model of pulsatile flow there would be only one value of y_G, and only one set of artifacts with occurances every y_G in the phase encoding direction. In reality there would be a distribution of fluctuating velocity components that depended on the nature of the motion. These would give a distribution of y_G values and, as a result, artifacts at many different spacings in the final two dimensional image complex data set and subsequent images (see Chapter 6).

Note that, if there are no varying velocity components, i.e., v_z is constant, then the $\delta(y - my_G)$ term would not be present. All that would remain is the phase term exp(iψ). This is consistant with the argument earlier in section 3.2.

Appendix C

In this appendix the Bloch equation is transformed into the rotating frame of reference. An iterative Runge Kutta solution to the equation in this reference frame is presented. Equation 2.33) gives the form of the Bloch equation in the laboratory frame of reference. It is usual to consider NMR experiments in the rotating frame of reference. The equations describing the motion of the magnetization are much simplified in this reference frame. The basis for the following mathematical development can be found in Mansfield and Morris (1982).

The magnetic field, $\underline{B}$, in the equation is given by

$$\underline{B} = \underline{k}B_0' + \underline{B}_1(t) \qquad \text{C.1)}$$

where

$$B_0' = B_0 + \underline{G}_r \cdot \underline{r}$$

B_0 is the magnitude of the main, static magnetic field and $\underline{G}_r$ is a linear magnetic field gradient vector. $\underline{r}$ is the position vector. $\underline{B}_1(t)$ is a radio frequency (RF) magnetic field rotating with angular frequency ω near or at the Larmor frequency, and has the form

$$\underline{B}_1(t) = \underline{i}B_1\cos\omega t - \underline{j}B_1\sin\omega t \qquad \text{C.2)}$$

This is equivalent to B_1 oriented along the x axis of the rotating frame. All other terms in equation 2.33) are as defined in Chapter 2. Remembering that

$$\underline{u}\times\underline{v} = \begin{vmatrix} u_2 & u_3 \\ v_2 & v_3 \end{vmatrix}\underline{i} - \begin{vmatrix} u_1 & u_3 \\ v_1 & v_3 \end{vmatrix}\underline{j} + \begin{vmatrix} u_1 & u_2 \\ v_1 & v_2 \end{vmatrix}\underline{k},$$

where the vertical bars represent the determinant of the matrix, the Bloch equations in the laboratory frame of reference can be written as

$$d\underline{M}/dt = \underline{M}\begin{bmatrix} -1/T_2 & -\gamma B_0' & -\gamma B_1\sin\omega t \\ \gamma B_0' & -1/T_2 & -\gamma B_1\cos\omega t \\ \gamma B_1\sin\omega t & \gamma B_1\cos\omega t & -1/T_1 \end{bmatrix} + \underline{M}_0/T_1$$

$$= \underline{M}\,\Re + \underline{M}_0/T_1 \qquad \text{C.3)}$$

where $\underline{M} = [\, \underline{i}M_x \;\; \underline{j}M_y \;\; \underline{k}M_z \,]$, a row vector representation of $\underline{M}$.

The transformation to the rotating frame begins with examination of the motion of the rotating frame axes in the laboratory frame of reference (see Figures C.1 and C.2). The matrix to translate from the laboratory frame to the rotating frame is

$$U^{-1} = \begin{bmatrix} \cos\omega t & \sin\omega t & 0 \\ -\sin\omega t & \cos\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \text{C.4)}$$

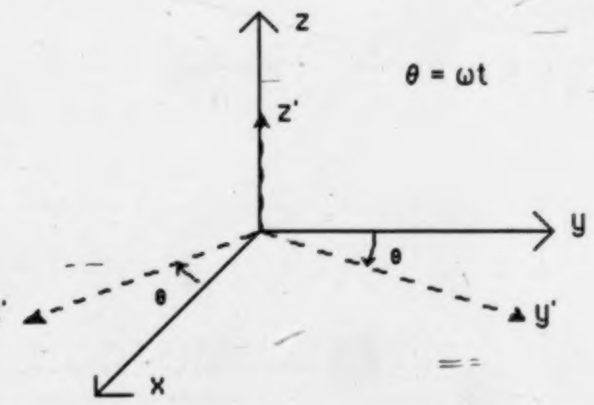


Figure C.1 A set of cartesian axes (x' y' z') rotating about the laboratory frame of reference.

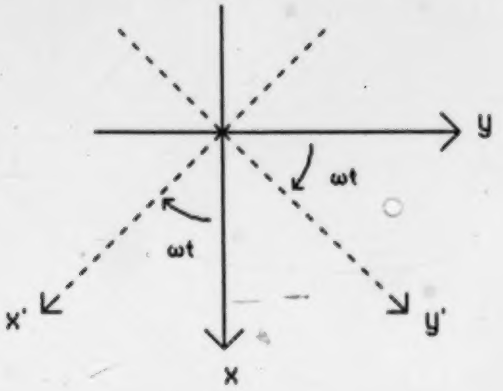


Figure C.2 The rotating frame axes (x' y') with respect to the laboratory frame axes. The translation matrix U^{-1} relates the rotating axes to the laboratory frame axes.

Equation C.3) can be written as

$$\begin{aligned}(d\underline{M}/dt)U^{-1} &= \underline{M}\ \Re\ U^{-1} + \underline{M}_0\ U^{-1}/T_1 \\ &= \underline{M}\ U^{-1}\ U\ \Re\ U^{-1} + \underline{M}_0\ U^{-1}/T_1 \qquad \text{C.5)}\end{aligned}$$

The change in time of the magnetization in the rotating frame is $d(\underline{M}\ U^{-1})/dt$. Remembering the chain rule of differentiation,

$$\frac{d(\underline{M}\ U^{-1})}{dt} = \frac{d\underline{M}}{dt}U^{-1} + \underline{M}\frac{dU^{-1}}{dt} \qquad \text{C.6)}$$

equation C.5) becomes

$$\begin{aligned}\frac{d(\underline{M}\ U^{-1})}{dt} &= \underline{M}\ U^{-1}\ U\ \Re\ U^{-1} + \underline{M}\frac{dU^{-1}}{dt} + \frac{\underline{M}_0\ U^{-1}}{T_1} \\ &= \underline{M}\ U^{-1}\left\{U\ \Re\ U^{-1} + U\frac{dU^{-1}}{dt}\right\} + \frac{\underline{M}_0\ U^{-1}}{T_1}\end{aligned}$$

C.7)

Using $\underline{M}$ as the rotating frame magnetization vector ($\underline{\mathit{M}}$ U^{-1}) and $\underline{M}_0$ as the rotating frame thermal equilibrium magnetization vector, equation C.7) is written as

$$\frac{d\underline{M}}{dt} = \underline{M}\,R + \frac{\underline{M}_0}{T_1}\,.$$

C.8)

where

$$R = U\,\mathfrak{R}\,U^{-1} + U\frac{dU^{-1}}{dt}$$

The inverse matrix, U, of U^{-1} is given by

$$U = \text{cofactor}[\,U^{-1}\,]/|\,U^{-1}\,|$$

$$\text{cofactor}[\,U^{-1}\,] = \begin{bmatrix} \cos\omega t & -\sin\omega t & 0 \\ \sin\omega t & \cos\omega t & 0 \\ 0 & 0 & \cos\omega^2 t + \sin\omega^2 t \end{bmatrix}$$

$$|\,U^{-1}\,| = \cos^2\omega t + \sin^2\omega t = 1$$

Thus,

$$U = \begin{bmatrix} \cos\omega t & -\sin\omega t & 0 \\ \sin\omega t & \cos\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Solving for the first term of R in equation C.8)

$$U \Re U^{-1} = \begin{bmatrix} \cos\omega t & -\sin\omega t & 0 \\ \sin\omega t & \cos\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1/T_2 & -\gamma B_0' & -\gamma B_1 \sin\omega t \\ \gamma B_0' & -1/T_2 & -\gamma B_1 \cos\omega t \\ \gamma B_1 \sin\omega t & \gamma B_1 \cos\omega t & -1/T_1 \end{bmatrix}$$

$$\cdot \begin{bmatrix} \cos\omega t & \sin\omega t & 0 \\ -\sin\omega t & \cos\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} -1/T_2 & -\gamma B_0' & 0 \\ \gamma B_0' & -1/T_2 & -\gamma B_1 \\ 0 & \gamma B_1 & -1/T_1 \end{bmatrix}$$

The second term of R is

$$U \frac{dU^{-1}}{dt} = \begin{bmatrix} \cos\omega t & -\sin\omega t & 0 \\ \sin\omega t & \cos\omega t & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -\omega \sin\omega t & \omega \cos\omega t & 0 \\ -\omega \cos\omega t & -\omega \sin\omega t & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & \omega & 0 \\ -\omega & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Using the results from above, equation C.8) is now

$$d\underline{M}/dt = \underline{M}\begin{bmatrix} -1/T_2 & -\gamma B_0' + \omega & 0 \\ \gamma B_0' - \omega & -1/T_2 & -\gamma B_1 \\ 0 & \gamma B_1 & -1/T_1 \end{bmatrix} + \underline{M}_0/T_1$$

Letting

$$\Delta\omega = \gamma B_0' - \omega$$

$$\omega_1 = \gamma B_1$$

$$d\underline{M}/dt = \underline{M}\begin{bmatrix} -1/T_2 & -\Delta\omega & 0 \\ \Delta\omega & -1/T_2 & -\omega_1 \\ 0 & \omega_1 & -1/T_1 \end{bmatrix} + \underline{M}_0/T_1$$

Thus, for $\underline{B}_1$ oriented along the positive x axis of the rotating frame, the Bloch equations in the rotating frame are

$$\begin{aligned} dM_x/dt &= \Delta\omega M_y - M_x/T_2 \\ dM_y/dt &= -\Delta\omega M_x + \omega_1 M_z - M_y/T_2 \\ dM_z/dt &= -\omega_1 M_y + (M_0 - M_z)/T_1 \end{aligned} \qquad \text{C.9)}$$

In a similar fashion, the equations with the $\underline{B}_1$ field along the negative x axis and the positive and negative y axes of the rotating frame can be derived. All that need be changed is the orientation of $\underline{B}_1$ at t=0 in equation C.2) (i.e., at t=0 it is along $-\underline{i}$, $\underline{j}$ or $-\underline{j}$). The equations in C.9) are seen to be a system of three first order first degree differential equations. There are many different numerical methods to solve this type of equation, but only the fourth order Runge Kutta method will be considered here (Burden et al., 1981).

Runge Kutta techniques are based on expansions of the Taylor series with the number of Taylor series terms reflected in the order of the particular technique. Defining ξ_i as the dependent variable and ζ_i as the independent variable of the ith differential equation of m equations, the technique proceeds as follows. The region from the initial value point, a, to the point of the desired solution, b, is divided into n subintervals. Over each subinterval in the region, the following iteration is performed j = 1 to n times.

$$\xi_{i,j+1} = \xi_{i,j} + (1/6)(k_{1,i} + 2k_{2,i} + 2k_{3,i} + k_{4,i}) \qquad \text{for each } i = 1, 2, \ldots, m \qquad \text{C.10)}$$

where

$$k_{1,i} = f_i(\zeta_j, \xi_{1,j}, \xi_{2,j}, \ldots, \xi_{m,j})h \qquad \text{for each } i = 1, 2, \ldots, m$$

$$k_{2,i} = f_i(\zeta_j + 1/2h, \xi_{1,j} + 1/2k_{1,1}, \xi_{2,j} + 1/2k_{1,2}, \ldots, \xi_{m,j} + 1/2k_{1,m})h \qquad \text{for each } i = 1, 2, \ldots, m$$

$$k_{3,i} = f_i(\zeta_j + 1/2h, \xi_{1,j} + 1/2k_{2,1}, \xi_{2,j} + 1/2k_{2,2}, \ldots, \xi_{m,j} + 1/2k_{2,m})h \qquad \text{for each } i = 1, 2, \ldots, m$$

$$k_{4,i} = f_i(\zeta_j + h, \xi_{1,j} + k_{3,1}, \xi_{2,j} + k_{3,2}, \ldots, \xi_{m,j} + k_{3,m})h \qquad \text{for each } i = 1, 2, \ldots, m$$

$$h = (b - a)/n \qquad \text{C.11)}$$

The error in this technique over the region is on the order of h^4 (Burden et al., 1981). For the system of equations C.9), the fourth order Runge Kutta solution for one time slice τ would be

$$h = \tau/n$$

$$k_{1,x} = (\Delta\omega\xi_{y,j} - \xi_{x,j}/T_2)h$$

$$k_{1,y} = (-\Delta\omega\xi_{x,j} + \omega_1\xi_{z,j} - \xi_{y,j}/T_2)h$$

$$k_{1,z} = (-\omega_1\xi_{y,j} + (M_0 - \xi_{z,j})/T_1)h$$

$$k_{2,x} = (\Delta\omega(\xi_{y,j} + .5k_{1,y}) - (\xi_{x,j} + .5k_{1,x})/T_2)h$$

$$k_{2,y} = (-\Delta\omega(\xi_{x,j} + .5k_{1,x}) + \omega_1(\xi_{z,j} + .5k_{1,z}) - (\xi_{y,j} + .5k_{1,y})/T_2)h$$

$$k_{2,z} = (-\omega_1(\xi_{y,j} + .5k_{1,y}) + (M_0 - \xi_{z,j} - .5k_{1,z})/T_1)h$$

$$k_{3,x} = (\Delta\omega(\xi_{y,j} + .5k_{2,y}) - (\xi_{x,j} + .5k_{2,x})/T_2)h$$

$$k_{3,y} = (-\Delta\omega(\xi_{x,j} + .5k_{2,x}) + \omega_1(\xi_{z,j} + .5k_{2,z}) - (\xi_{y,j} + .5k_{2,y})/T_2)h$$

$$k_{3,z} = (-\omega_1(\xi_{y,j} + .5k_{2,y}) + (M_0 - \xi_{z,j} - .5k_{2,z})/T_1)h$$

$$k_{4,x} = (\Delta\omega(\xi_{y,j} + k_{3,y}) - (\xi_{x,j} + k_{3,x})/T_2)h$$

$$k_{4,y} = (-\Delta\omega(\xi_{x,j} + k_{3,x}) + \omega_1(\xi_{z,j} + k_{3,z}) - (\xi_{y,j} + k_{3,y})/T_2)h$$

$$k_{4,z} = (-\omega_1(\xi_{y,j} + k_{3,y}) + (M_0 - \xi_{z,j} - k_{3,z})/T_1)h$$

$$\xi_{x,j+1} = \xi_{x,j} + (1/6)(k_{1,x} + 2k_{2,x} + 2k_{3,x} + k_{4,x})$$

$$\xi_{y,j+1} = \xi_{y,j} + (1/6)(k_{1,y} + 2k_{2,y} + 2k_{3,y} + k_{4,y})$$

$$\xi_{z,j+1} = \xi_{z,j} + (1/6)(k_{1,z} + 2k_{2,z} + 2k_{3,z} + k_{4,z})$$

The initial conditions would be

$$\xi_{x,1} = M_x$$

$$\xi_{y,1} = M_y$$

$$\xi_{z,1} = M_z$$

Following n iterations,

$$M_x = \xi_{x,n}$$
$$M_y = \xi_{y,n}$$
$$M_z = \xi_{z,n}$$

would give the values of M_x, M_y and M_z τ seconds later.

The form of the programmed solution varies slightly from this in an effort to reduce the number of computations and hence, overall simulation running time.

Appendix D

A listing of the simulation package modules plus two " INCLUDE " modules, BLOCHD1D.FOR and BLOCKSTK.FOR, at the end.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
ccccccc
C
        SUBROUTINE DM_DT_PLUSX
C
C  A routine based on Mansfield and Morris's solution of the Bloch equations.
C  ( NMR Imaging in Biomedicine, Academic Press, 1982 )
C
C  These equations are solved for the RF field applied along the positive
C  x-axis in the rotating frame.
C
C  A Runge Kutta ( 4th order ) solution is used. It can be found in most
C  books on numerical analysis.
C
C       Global Variables ( as found in BLOCHD1D.FOR ):
C
C       M                ----> a N+1 dimensional array of magnetization
C                              components in N space. The first N indices
C                              correspond to the coordinates of the 'spin'
C                              in space. The last index indicates the
C                              directional vector. i.e.
C                                    M(x,y,z,d) represents 'spin' x,y,z and
C                                          component d of the spin.
C                                             d = 1 is the x component
C                                               =2 "  " y   "
C                                               =3 "  " z   "
C
C                              The bounds on the N indices are declared
C                              in BLOCHD1D.FOR
C
C       GRAD             ----> a N dimensional array of magnetic field
C                              gradient values. The N indices
C                              define the point value of the gradient
C                              ( along the z-axis of course ) at the
C                              corresponding N-space location in array M.
C
C       H1               ----> a real value for the current H1 ( or B1 ) field
C
C       GAMMA            ----> a real constant that is set to the gyromagneto
C                              ratio for the spin species.
C
C       T1A              ----> a N dimensional array containing the T1
C                              values for the corresponding magnetization
C                              vectors in array M.
C
C       T2A              ----> a N dimensional array containing the T2
C                              values for the corresponding magnetization
C                              vectors in array M.
C
C       TINC             ----> is a real value for the incremental time step
C
C       N                ----> is an integer value for the number of steps
```

```
C                               to take in the Runge Kutta approximation.
C
C       .
C       .
C       .
C       XDIM
C       YDIM
C       ZDIM            ----> integer values corresponding to the dimensions
C                             of the spin array in the experiment ( see
C                             entry for M array above )
C
C 4th order Runge Kutta routine to calculate dMX/dt
C
        INCLUDE 'BLOCHD1D.FOR'
C
C loop through the array
C
        DO Z = 1,ZDIM
C
        ZMX = M(Z,1)
        ZMY = M(Z,2)
        ZMZ = M(Z,3)
        T1 = T1A(Z)
        T2 = T2A(Z)
        ZW = TINC/FLOAT(N)
C
C 4th order Runge Kutta routine to calculate dMX/dt, d/MY/dt, dMZ/dt
C
        ZDEL_OMEGA = GAMMA*GRAD(Z)
        ZOMEGA1 = H1 * GAMMA
        DO I = 1, N
C
C do the K1s
C
          ZK1X = ( ZDEL_OMEGA*ZMY - ZMX/T2 )*ZW
          ZK1Y = ( -1.*ZDEL_OMEGA*ZMX + ZOMEGA1*ZMZ - ZMY/T2 )*ZW
          ZK1Z = ( -1.*ZOMEGA1*ZMY + (M0 - ZMZ)/T1 )*ZW
C
C do the K2s
C
          ZK2X = ( ZDEL_OMEGA*(ZMY+.5*ZK1Y) - (ZMX+.5*ZK1X)/T2 )*ZW
          ZK2Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK1X)
     2             + ZOMEGA1*(ZMZ+.5*ZK1Z) - (ZMY+.5*ZK1Y)/T2 )*ZW
          ZK2Z = ( -1.*ZOMEGA1*(ZMY+.5*ZK1Y)
     2             + (M0 - ZMZ - .5*ZK1Z)/T1 )*ZW
C
C do the K3s
C
          ZK3X = ( ZDEL_OMEGA*(ZMY+.5*ZK2Y) - (ZMX+.5*ZK2X)/T2 )*ZW
          ZK3Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK2X)
     3             + ZOMEGA1*(ZMZ+.5*ZK2Z) - (ZMY+.5*ZK2Y)/T2 )*ZW
          ZK3Z = ( -1.*ZOMEGA1*(ZMY+.5*ZK2Y)
     3             + (M0 - ZMZ - .5*ZK2Z)/T1 )*ZW
C
C do the K4s
C
          ZK4X = ( ZDEL_OMEGA*(ZMY+ZK3Y) - (ZMX+ZK3X)/T2 )*ZW
          ZK4Y = ( -1.*ZDEL_OMEGA*(ZMX+ZK3X)
     4             + ZOMEGA1*(ZMZ+ZK3Z) - (ZMY+ZK3Y)/T2 )*ZW
          ZK4Z = ( -1.*ZOMEGA1*(ZMY+ZK3Y)
     4             + (M0 - ZMZ - ZK3Z)/T1 )*ZW
C
C and finally the next approximation
C
```

```
        ZMX = ZMX + 1./6.*(ZK1X+ZK2X+ZK2X+ZK3X+ZK3X+ZK4X)
        ZMY = ZMY + 1./6.*(ZK1Y+ZK2Y+ZK2Y+ZK3Y+ZK3Y+ZK4Y)
        ZMZ = ZMZ + 1./6.*(ZK1Z+ZK2Z+ZK2Z+ZK3Z+ZK3Z+ZK4Z)
      END DO
C
C set the array values back
C
      M(Z,1) = ZMX
      M(Z,2) = ZMY
      M(Z,3) = ZMZ
C
      END DO IZ LOOP
C
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCC
C
      SUBROUTINE DM_DT_MINUSX
C
C  A routine based on Mansfield and Morris's solution of the Bloch equations.
C  ( NMR Imaging in Biomedicine, Academic Press, 1982 )
C
C  These equations are solved for the RF field applied along the negative
C  x-axis in the rotating frame.
C
C  A Runge Kutta ( 4th order ) solution is used. It can be found in most
C  books on numerical analysis.
C
C     Global Variables ( as found in BLOCHD1D.FOR ):
C
C     M              ----> a N+1 dimensional array of magnetization
C                          components in N space. The first N indices
C                          correspond to the coordinates of the 'spin'
C                          in space. The last index indicates the
C                          directional vector. i.e.
C                              M(x,y,z,d) represents 'spin' x,y,z and
C                                   component d of the spin.
C                                       d = 1 is the x component
C                                         =2 " " y  "
C                                         =3 " " z  "
C
C                          The bounds on the N indices are declared
C                          in BLOCHD1D.FOR
C
C     GRAD           ----> a N dimensional array of magnetic field
C                          gradient values. The N indices
C                          define the point value of the gradient
C                          ( along the z-axis of course ) at the
C                          corresponding N-space location in array M.
C
C     H1             ----> a real value for the current H1 ( or B1 ) field
C
C     GAMMA          ----> a real constant that is set to the gyromagneto
C                          ratio for the spin species.
C
C     T1A            ----> a N dimensional array containing the T1
C                          values for the corresponding magnetization
C                          vectors in array M.
C
C     T2A            ----> a N dimensional array containing the T2
C                          values for the corresponding magnetization
C                          vectors in array M.
C
```

```
C        TINC               ----> is a real value for the incremental time step
C
C        N                  ----> is an integer value for the number of steps
C                                 to take in the Runge Kutta approximation.
C
C        .
C        .
C        .
C        XDIM
C        YDIM
C        ZDIM               ----> integer values corresponding to the dimensions
C                                 of the spin array in the experiment ( see
C                                 entry for M array above )
C
C 4th order Runge Kutta routine to calculate dMX/dt
C
         INCLUDE 'BLOCHD1D.FOR'
C
C loop through the array
C
         DO Z = 1,ZDIM
C
         ZMX = M(Z,1)
         ZMY = M(Z,2)
         ZMZ = M(Z,3)
         T1 = T1A(Z)
         T2 = T2A(Z)
         ZW = TINC/FLOAT(N)
C
C 4th order Runge Kutta routine to calculate dMX/dt, d/MY/dt, dMZ/dt
C
         ZDEL_OMEGA = GAMMA*GRAD(Z)
         ZOMEGA1 = H1 * GAMMA
         DO I = 1, N
C
C do the K1s
C
           ZK1X = ( ZDEL_OMEGA*ZMY - ZMX/T2 )*ZW
           ZK1Y = ( -1.*ZDEL_OMEGA*ZMX - ZOMEGA1*ZMZ - ZMY/T2 )*ZW
           ZK1Z = ( ZOMEGA1*ZMY + (M0 - ZMZ)/T1 )*ZW
C
C do the K2s
C
           ZK2X = ( ZDEL_OMEGA*(ZMY+.5*ZK1Y) - (ZMX+.5*ZK1X)/T2 )*ZW
           ZK2Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK1X)
     2             - ZOMEGA1*(ZMZ+.5*ZK1Z) - (ZMY+.5*ZK1Y)/T2 )*ZW
           ZK2Z = ( ZOMEGA1*(ZMY+.5*ZK1Y)
     2             + (M0 - ZMZ - .5*ZK1Z)/T1 )*ZW
C
C do the K3s
C
           ZK3X = ( ZDEL_OMEGA*(ZMY+.5*ZK2Y) - (ZMX+.5*ZK2X)/T2 )*ZW
           ZK3Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK2X)
     3             - ZOMEGA1*(ZMZ+.5*ZK2Z) - (ZMY+.5*ZK2Y)/T2 )*ZW
           ZK3Z = ( ZOMEGA1*(ZMY+.5*ZK2Y)
     3             + (M0 - ZMZ - .5*ZK2Z)/T1 )*ZW
C
C do the K4s
C
           ZK4X = ( ZDEL_OMEGA*(ZMY+ZK3Y) - (ZMX+ZK3X)/T2 )*ZW
           ZK4Y = ( -1.*ZDEL_OMEGA*(ZMX+ZK3X)
     4             - ZOMEGA1*(ZMZ+ZK3Z) - (ZMY+ZK3Y)/T2 )*ZW
           ZK4Z = ( ZOMEGA1*(ZMY+ZK3Y)
     4             + (M0 - ZMZ - ZK3Z)/T1 )*ZW
```

```
C
C and finally the next approximation
C
            ZMX = ZMX + 1./6.*(ZK1X+ZK2X+ZK2X+ZK3X+ZK3X+ZK4X)
            ZMY = ZMY + 1./6.*(ZK1Y+ZK2Y+ZK2Y+ZK3Y+ZK3Y+ZK4Y)
            ZMZ = ZMZ + 1./6.*(ZK1Z+ZK2Z+ZK2Z+ZK3Z+ZK3Z+ZK4Z)
          END DO
C
C set the array values back
C
        M(Z,1) = ZMX
        M(Z,2) = ZMY
        M(Z,3) = ZMZ
C
        END DO IZ LOOP
C
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
        SUBROUTINE DM_DT_PLUSY
C
C  A routine based on Mansfield and Morris's solution of the Bloch equations.
C  ( NMR Imaging in Biomedicine, Academic Press, 1982 )
C
C  These equations are solved for the RF field applied along the positive
C  y-axis in the rotating frame.
C
C  A Runge Kutta ( 4th order ) solution is used. It can be found in most
C  books on numerical analysis.
C
C       Global Variables ( as found in BLOCHD1D.FOR ):
C
C       M                   ----> a N+1 dimensional array of magnetization
C                                 components in N space. The first N indices
C                                 correspond to the coordinates of the 'spin'
C                                 in space. The last index indicates the
C                                 directional vector. i.e.
C                                       M(x,y,z,d) represents 'spin' x,y,z and
C                                             component d of the spin.
C                                                  d = 1 is the x component
C                                                    = 2 "  " y     "
C                                                    = 3 "  " z     "
C
C                                 The bounds on the N indices are declared
C                                 in BLOCHD1D.FOR
C
C       GRAD                ----> a N dimensional array of magnetic field
C                                 gradient values. The N indices
C                                 define the point value of the gradient
C                                 ( along the z-axis of course ) at the
C                                 corresponding N-space location in array M.
C
C       H1                  ----> a real value for the current H1 ( or B1 ) field
C
C       GAMMA               ----> a real constant that is set to the gyromagneto
C                                 ratio for the spin species.
C
C       T1A                 ----> a N dimensional array containing the T1
C                                 values for the corresponding magnetization
C                                 vectors in array M.
C
```

```
C        T2A              ----> a N dimensional array containing the T2
C                               values for the corresponding magnetization
C                               vectors in array M.
C
C        TINC             ----> is a real value for the incremental time step
C
C        N                ----> is an integer value for the number of steps
C                               to take in the Runge Kutta approximation.
C
C        .
C        .
C        .
C        XDIM
C        YDIM
C        ZDIM             ----> integer values corresponding to the dimensions
C                               of the spin array in the experiment ( see
C                               entry for M array above )
C
C 4th order Runge Kutta routine to calculate dMX/dt
C
         INCLUDE 'BLOCHD1D.FOR'
C
C loop through the array
C
         DO Z = 1,ZDIM
C
         ZMX = M(Z,1)
         ZMY = M(Z,2)
         ZMZ = M(Z,3)
         T1 = T1A(Z)
         T2 = T2A(Z)
         ZW = TINC/FLOAT(N)
C
C 4th order Runge Kutta routine to calculate dMX/dt, d/MY/dt, dMZ/dt
C
         ZDEL_OMEGA = GAMMA*GRAD(Z)
         ZOMEGA1 = H1 * GAMMA
         DO I = 1, N
C
C do the K1s
C
           ZK1X = ( ZDEL_OMEGA*ZMY - ZOMEGA1*ZMZ - ZMX/T2 )*ZW
           ZK1Y = ( -1.*ZDEL_OMEGA*ZMX - ZMY/T2 )*ZW
           ZK1Z = ( ZOMEGA1*ZMX + (M0 - ZMZ)/T1 )*ZW
C
C do the K2s
C
           ZK2X = ( ZDEL_OMEGA*(ZMY+.5*ZK1Y)
     2             - ZOMEGA1*(ZMZ+.5*ZK1Z) - (ZMX+.5*ZK1X)/T2 )*ZW
           ZK2Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK1X) - (ZMY+.5*ZK1Y)/T2 )*ZW
           ZK2Z = ( ZOMEGA1*(ZMX+.5*ZK1X)
     2             + (M0 - ZMZ - .5*ZK1Z)/T1 )*ZW
C
C do the K3s
C
           ZK3X = ( ZDEL_OMEGA*(ZMY+.5*ZK2Y)
     3             - ZOMEGA1*(ZMZ+.5*ZK2Z) - (ZMX+.5*ZK2X)/T2 )*ZW
           ZK3Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK2X) - (ZMY+.5*ZK2Y)/T2 )*ZW
           ZK3Z = ( ZOMEGA1*(ZMX+.5*ZK2X)
     3             + (M0 - ZMZ - .5*ZK2Z)/T1 )*ZW
C
C do the K4s
C
           ZK4X = ( ZDEL_OMEGA*(ZMY+ZK3Y)
     4            - ZOMEGA1*(ZMZ+ZK3Z) - (ZMX+ZK3X)/T2 )*ZW
        ZK4Y = ( -1.*ZDEL_OMEGA*(ZMX+ZK3X) - (ZMY+ZK3Y)/T2 )*ZW
        ZK4Z = ( ZOMEGA1*(ZMX+ZK3X)
     4            + (M0 - ZMZ - ZK3Z)/T1 )*ZW
C
C and finally the next approximation
C
        ZMX = ZMX + 1./6.*(ZK1X+ZK2X+ZK2X+ZK3X+ZK3X+ZK4X)
        ZMY = ZMY + 1./6.*(ZK1Y+ZK2Y+ZK2Y+ZK3Y+ZK3Y+ZK4Y)
        ZMZ = ZMZ + 1./6.*(ZK1Z+ZK2Z+ZK2Z+ZK3Z+ZK3Z+ZK4Z)
      END DO
C
C set the array values back
C
      M(Z,1) = ZMX
      M(Z,2) = ZMY
      M(Z,3) = ZMZ
C
      END DO IZ LOOP
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE DM_DT_MINUSY
C
C  A routine based on Mansfield and Morris's solution of the Bloch equations.
C  ( NMR Imaging in Biomedicine, Academic Press, 1982 )
C
C  These equations are solved for the RF field applied along the negative
C  y-axis in the rotating frame.
C
C  A Runge Kutta ( 4th order ) solution is used. It can be found in most
C  books on numerical analysis.
C
C      Global Variables ( as found in BLOCHD1D.FOR ):
C
C      M                ---> a N+1 dimensional array of magnetization
C                            components in N space. The first N indices
C                            correspond to the coordinates of the 'spin'
C                            in space. The last index indicates the
C                            directional vector. i.e.
C                                  M(x,y,z,d) represents 'spin' x,y,z and
C                                        component d of the spin.
C                                              d = 1 is the x component
C                                               =2 "  " y   "
C                                               =3 "  " z   "
C
C                            The bounds on the N indices are declared
C                            in BLOCHD1D.FOR
C
C      GRAD             ---> a N dimensional array of magnetic field
C                            gradient values. The N indices
C                            define the point value of the gradient
C                            ( along the z-axis of course ) at the
C                            corresponding N-space location in array M.
C
C      H1               ---> a real value for the current H1 ( or B1 ) field
C
C      GAMMA            ---> a real constant that is set to the gyromagneto
C                            ratio for the spin species.
C
```

```
C         T1A                ----> a N dimensional array containing the T1
C                                  values for the corresponding magnetization
C                                  vectors in array M.
C
C         T2A                ----> a N dimensional array containing the T2
C                                  values for the corresponding magnetization
C                                  vectors in array M.
C
C         TINC               ----> is a real value for the incremental time step
C
C         N                  ----> is an integer value for the number of steps
C                                  to take in the Runge Kutta approximation.
C
C         .
C         .
C         .
C         XDIM
C         YDIM
C         ZDIM               ----> integer values corresponding to the dimensions
C                                  of the spin array in the experiment ( see
C                                  entry for M array above )
C
C 4th order Runge Kutta routine to calculate dMX/dt
C
          INCLUDE 'BLOCHD1D.FOR'
C
C loop through the array
C
          DO Z = 1,ZDIM
C
          ZMX = M(Z,1)
          ZMY = M(Z,2)
          ZMZ = M(Z,3)
          T1 = T1A(Z)
          T2 = T2A(Z)
          ZW = TINC/FLOAT(N)
C
C 4th order Runge Kutta routine to calculate dMX/dt, d/MY/dt, dMZ/dt
C
          ZDEL_OMEGA = GAMMA*GRAD(Z)
          ZOMEGA1 = H1 * GAMMA
          DO I = 1, N
C
C do the K1s
C
            ZK1X = ( ZDEL_OMEGA*ZMY + ZOMEGA1*ZMZ - ZMX/T2 )*ZW
            ZK1Y = ( -1.*ZDEL_OMEGA*ZMX - ZMY/T2 )*ZW
            ZK1Z = ( -1.*ZOMEGA1*ZMX + (M0 - ZMZ)/T1 )*ZW
C
C do the K2s
C
            ZK2X = ( ZDEL_OMEGA*(ZMY+.5*ZK1Y)
          2            + ZOMEGA1*(ZMZ+.5*ZK1Z) - (ZMX+.5*ZK1X)/T2 )*ZW
            ZK2Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK1X) - (ZMY+.5*ZK1Y)/T2 )*ZW
            ZK2Z = ( -1.*ZOMEGA1*(ZMX+.5*ZK1X)
          2            + (M0 - ZMZ - .5*ZK1Z)/T1 )*ZW
C
C do the K3s
C
            ZK3X = ( ZDEL_OMEGA*(ZMY+.5*ZK2Y)
          3            + ZOMEGA1*(ZMZ+.5*ZK2Z) - (ZMX+.5*ZK2X)/T2 )*ZW
            ZK3Y = ( -1.*ZDEL_OMEGA*(ZMX+.5*ZK2X) - (ZMY+.5*ZK2Y)/T2 )*ZW
            ZK3Z = ( -1.*ZOMEGA1*(ZMX+.5*ZK2X)
          3            + (M0 - ZMZ - .5*ZK2Z)/T1 )*ZW
```

```
C
C do the K4s
C
            ZK4X = ( ZDEL_OMEGA*(ZMY+ZK3Y)
         4           + ZOMEGA1*(ZMZ+ZK3Z) - (ZMX+ZK3X)/T2 )*ZW
            ZK4Y = ( -1.*ZDEL_OMEGA*(ZMX+ZK3X) - (ZMY+ZK3Y)/T2 )*ZW
            ZK4Z = ( -1.*ZOMEGA1*(ZMX+ZK3X)
         4           + (M0 - ZMZ - ZK3Z)/T1 )*ZW
C
C and finally the next approximation
C
            ZMX = ZMX + 1./6.*(ZK1X+ZK2X+ZK2X+ZK3X+ZK3X+ZK4X)
            ZMY = ZMY + 1./6.*(ZK1Y+ZK2Y+ZK2Y+ZK3Y+ZK3Y+ZK4Y)
            ZMZ = ZMZ + 1./6.*(ZK1Z+ZK2Z+ZK2Z+ZK3Z+ZK3Z+ZK4Z)
         END DO
C
C set the array values back
C
         M(Z,1) = ZMX
         M(Z,2) = ZMY
         M(Z,3) = ZMZ
C
         END DO IZ LOOP
C
         RETURN
         END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
         SUBROUTINE NO_RF_GRADIENTS_ON_IDEAL
C
C This routine calculates what the components of the magnetic moment should
C be along each axis in N-space when there is ** NO ** r.f. energy input
C to the system and there are gradients on !!!
C
C This routine may be used after any pulse event.
C
C The equations describing this are given in "NMR Imaging In Biomedicine"
C by Mansfield and Morris.
C
C This routine only approximates the results of having the gradients on for
C a time ( TINC ), i.e. the Bloch Eqns. are not solved, only the intuitive
C solutions ( what one suspects should happen ) are used
C
C
C        Global variables:
C
C        T                ---> a real value for the time elapse in the pulse
C                              program ( or experiment if you like ).
C
C        TINC             ---> a real value for the time the spins are to
C                              be allowed to process under a gradient.
C
C        T1A              ---> a N dimensional array containing the T1
C                              values for the corresponding magnetization
C                              vectors in array M.
C
C        T2A              ---> a N dimensional array containing the T2
C                              values for the corresponding magnetization
C                              vectors in array M.
C
C        M                ---> a N+1 dimensional array of magnetization
C                              components in N space. The first N indices
```

```
C                                correspond to the coordinates of the 'spin'
C                                in space. The last index indicates the
C                                directional vector. i.e.
C                                    M(x,y,z,d) represents 'spin' x,y,z and
C                                         component d of the spin.
C                                              d = 1 is the x component
C                                               = 2 "  " y   "
C                                               = 3 "  " z   "
C
C                                The bounds on the N indices are declared
C                                in BLOCHD1D.FOR
C      .
C      .
C      .
C      XDIM
C      YDIM
C      ZDIM                  ----> integer values corresponding to the dimensions
C                                of the spin array in the experiment ( see
C                                entry for M array above )
C
      INCLUDE 'BLOCHD1D.FOR'
C
      ZPI = 3.1415926
C
      DO Z = 1,ZDIM
         T1 = T1A(Z)
         T2 = T2A(Z)
C
C determine the future value of Mz TINC seconds from now.
C
         M(Z,3) = M(Z,3) - (M(Z,3)-M0) *
     1                       ( 1.-EXP(-TINC/T1) )
C
C now find what the component in the xy plane will be in TINC seconds
C
         ZM_XY = SQRT( M(Z,1)**2 + M(Z,2)**2 )
         IF (ABS(M(Z,2)).LE.1.E-28) THEN
              IF(M(Z,1).NE.0.0) THEN
            ZTHETA = 3.14159265 / 2.0 * M(Z,1)/ABS(M(Z,1))
              ELSE
                ZTHETA = 3.14159265 / 2.0
              END IF
         ELSE
           ZTHETA = ATAN( M(Z,1)/M(Z,2) )
            IF(M(Z,2).LT.0.0.AND.M(Z,1).LT.0.0) THEN
                ZTHETA = ZTHETA - ZPI
              END IF
            IF(M(Z,2).LT.0.0.AND.M(Z,1).GT.0.0) THEN
                ZTHETA = ZTHETA + ZPI
              END IF
         ENDIF
         ZDEL_OMEGA = GAMMA*GRAD(Z)
C
C for Mx
C
         M(Z,1) = ZM_XY * SIN( ZDEL_OMEGA*TINC + ZTHETA )*
     1                    EXP(-TINC/T2)
C
C for My
C
         M(Z,2) = ZM_XY * COS( ZDEL_OMEGA*TINC + ZTHETA )*
     1                    EXP(-TINC/T2)
      END DO
C
```

```
C update global time T
C
      T = T + TINC
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE NO_RF_GRADIENTS_ON
C
C This routine calculates what the components of the magnetic moment should
C be along each axis ( X, Y, Z ) when there is ** NO ** r.f. energy input
C to the system and there are gradients on !!!
C
C This routine may be used after any pulse event.
C
C The equations describing this are given in "NMR Imaging In Biomedicine"
C by Mansfield and Morris, pg. 45,46 .
C
C
C     Global variables:
C
C     T             ----> a real value for the time elapse in the pulse
C                         program ( or experiment if you like ).
C
C     TINC          ----> a real value for the time the spins are to
C                         be allowed to precess under a gradient.
C
C     T1A           ----> a N dimensional array containing the T1
C                         values for the corresponding magnetization
C                         vectors in array M.
C
C     T2A           ----> a N dimensional array containing the T2
C                         values for the corresponding magnetization
C                         vectors in array M.
C
C     M             ----> a N+1 dimensional array of magnetization
C                         components in N space. The first N indices
C                         correspond to the coordinates of the 'spin'
C                         in space. The last index indicates the
C                         directional vector. i.e.
C                               M(x,y,z,d) represents 'spin' x,y,z and
C                                    component d of the spin.
C                                         d = 1 is the x component
C                                           = 2 " " y    "
C                                           = 3 " " z    "
C
C                         The bounds on the N indices are declared
C                         in BLOCHD1D.FOR
C
C     .
C     .
C     .
C     XDIM
C     YDIM
C     ZDIM          ----> integer values corresponding to the dimensions
C                         of the spin array in the experiment ( see
C                         entry for M array above )
C
      INCLUDE 'BLOCHD1D.FOR'
C
C set time limit, a temp time variable; save the global time increment
C and set TINC to the proper step size
```

```
C
      ZT = T
      ZLIMIT = T + TINC
      ZTINC = TINC
      DO WHILE(TINC.GT.MAX_TINC)
        TINC = TINC/2.
      END DO
C
      DO WHILE(ZT.LT.ZLIMIT)
        CALL DM_DT_PLUSX
C
C update the time ZT
C
        ZT = ZT + TINC
        IF((ZLIMIT-ZT).LT.TINC) TINC = (ZLIMIT-ZT)
      END DO
C
C update global time T
C
      T = ZT
      TINC = ZTINC
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE NO_RF_NO_GRADIENTS
C
C This routine calculates what the components of the magnetic moment should
C be along each axis ( X, Y, Z ) when there is ** NO ** r.f. energy input
C to the system and there are ** NO ** gradients on !!!
C
C This routine may be used after any pulse event.
C
C The equations describing this are given in "NMR Imaging In Biomedicine"
C by Mansfield and Morris, pg. 45,46 .
C
C
C     Global variables:
C
C     T                 ----> a real value for the time elapse in the pulse
C                             program ( or experiment if you like ).
C
C     TINC              ----> a real value for the time the spins are to
C                             be allowed to process under a gradient.
C
C     T1A               ----> a N dimensional array containing the T1
C                             values for the corresponding magnetization
C                             vectors in array M.
C
C     T2A               ----> a N dimensional array containing the T2
C                             values for the corresponding magnetization
C                             vectors in array M.
C
C     M                 ----> a N+1 dimensional array of magnetization
C                             components in N space. The first N indices
C                             correspond to the coordinates of the 'spin'
C                             in space. The last index indicates the
C                             directional vector. i.e.
C                                  M(x,y,z,d) represents 'spin' x,y,z and
C                                       component d of the spin.
C                                            d = 1 is the x component
```

```
C                                        =2 " " y  "
C                                        =3 " " z  "
C
C                              The bounds on the N indices are declared
C                              in BLOCHD1D.FOR
C        .
C        .
C        .
C        XDIM
C        YDIM
C        ZDIM                ----> integer values corresponding to the dimensions
C                              of the spin array in the experiment ( see
C                              entry for M array above )
C
         INCLUDE 'BLOCHD1D.FOR'
C
         DO Z = 1,ZDIM
             T1 = T1A(Z)
             T2 = T2A(Z)
C
C determine the future value of Mz TINC seconds from now.
C
         M(Z,3) = M(Z,3) - (M(Z,3)-M0) * ( 1.-EXP(-TINC/T1) )
C
C now find what the component in the xy plane will be in TINC seconds
C
         ZM_XY = SQRT( M(Z,1)**2 + M(Z,2)**2 )
         IF (ABS(M(Z,2)).LE.1.E-28) THEN
           IF(M(Z,1).NE.0.0) THEN
             ZTHETA = 3.14159265 / 2.0 * M(Z,1)/ABS(M(Z,1))
           ELSE
             ZTHETA = 3.14159265 / 2.0
           END IF
         ELSE
           ZTHETA = ATAN( M(Z,1)/M(Z,2) )
         ENDIF
C
C for Mx
C
         M(Z,1) = ZM_XY * SIN( ZTHETA )*EXP(-TINC/T2)
C
C for My
C
         M(Z,2) = ZM_XY * COS( ZTHETA )*EXP(-TINC/T2)
C
         END DO
C
C adjust the time
C
         T = T + TINC
C
         RETURN
         END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC
C
         SUBROUTINE ACQUIRE_SIGNAL( ZDWELL )
C
C This routine uses again the BLOCH equations as solved for the situation
C of no r.f. energy being input to the system.
C
C The array of spins is sampled "N" times over a period "TINC". The samples
C are stored as complex points ( ala quadrature sensitive detection ) and
```

```
C in the following format;
C                         real == y-axis magnetization component
C                         imag == x-axis     "          "
C The results from one call are pushed onto a stack and should be placed
C in the final array TIME_DATA at the end of the simulated experiment.
C
C         Passed variables:
C
C         ZDWELL          ----> a real value for the sampling period
C
C         Global variables:
C
C         T               ----> a real value for the time elapse in the pulse
C                               program ( or experiment if you like ).
C
C         TINC            ----> a real value for the time the spins are to
C                               be allowed to precess under a gradient.
C
C         M               ----> a N+1 dimensional array of magnetization
C                               components in N space. The first N indices
C                               correspond to the coordinates of the 'spin'
C                               in space. The last index indicates the
C                               directional vector. i.e.
C                                   M(x,y,z,d) represents 'spin' x,y,z and
C                                       component d of the spin.
C                                           d = 1 is the x component
C                                             = 2 "  " y    "
C                                             = 3 "  " z    "
C
C                               The bounds on the N indices are declared
C                               in BLOCHD1D.FOR
C         .
C         .
C         .
C         XDIM
C         YDIM
C         ZDIM            ----> integer values corresponding to the dimensions
C                               of the spin array in the experiment ( see
C
          INCLUDE 'BLOCHD1D.FOR'
          INCLUDE 'BLOCHSTK.FOR'
C
C save the value of TINC
C
          ZTINC = TINC
          ZDEL_TINC = ZDWELL
          ZT = 0.0
          NUM_PUSHES = 0
          NPHASE = ZTINC/ZDWELL
C
C reset TINC for calls to NO_RF_GRADIENTS_ON
C
          TINC = ZDWELL
C
          CALL INIT_STACK
C
          DO WHILE(ZT.LT.ZTINC)
            ZT = ZT + ZDEL_TINC
            CALL NO_RF_GRADIENTS_ON
C
C now get the real and imaginary data
C
            ZREAL = 0.0
            ZIMAG = 0.0
            DO Z = 1,ZDIM
                  ZREAL = ZREAL + M(Z,2)
                  ZIMAG = ZIMAG + M(Z,1)
            END DO
            DATA = CMPLX( ZREAL, ZIMAG )
            CALL PUSH( DATA )
            NUM_PUSHES = NUM_PUSHES + 1
          END DO
C
C adjust the stack if necessary
C
          DO WHILE(NUM_PUSHES.LE.NPHASE)
            CALL PUSH( DATA )
            NUM_PUSHES = NUM_PUSHES + 1
          END DO
          TYPE *,'NUM_PUSHES = ',NUM_PUSHES
C
C restore TINC value but don't bother with global time adjustment as
C NO_RF_GRADIENTS_ON takes care of it.
C
          TINC = ZTINC
C
          RETURN
          END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC
C
          SUBROUTINE ACQUIRE_SIGNAL_IDEAL( ZDWELL )
C
C This routine uses again the BLOCH equations as solved for the situation
C of no r.f. energy being input to the system in the 'ideal' routine
C NO_RF_GRADIENTS_ON_IDEAL.
C
C The array of spins is sampled "N" times over a period "TINC". The samples
C are stored as complex points ( ala quadrature sensitive detection ) and
C in the following format;
C                   real == y-axis magnetization component
C                   imag == x-axis
C The results from one call are pushed onto a stack and should be placed
C in the final array TIME_DATA at the end of the simulated experiment.
C
C         Passed variables:
C
C         ZDWELL            ----> a real value for the sampling period
C
C         Global variables:
C
C         T                 ----> a real value for the time elapse in the pulse
C                                 program ( or experiment if you like ).
C
C         TINC              ----> a real value for the time the spins are to
C                                 be allowed to process under a gradient.
C
C         M                 ----> a N+1 dimensional array of magnetization
C                                 components in N space. The first N indices
C                                 correspond to the coordinates of the 'spin'
C                                 in space. The last index indicates the
C                                 directional vector. i.e.
C                                       M(x,y,z,d) represents 'spin' x,y,z and
C                                              component d of the spin.
C                                                 d = 1 is the x component
C                                                   = 2 " " y   "
C                                                   = 3 " " z   "
```

```
C
C                               The bounds on the N indices are declared
C                               in BLOCHD1D.FOR
C
C     .
C     .
C     XDIM
C     YDIM
C     ZDIM               ----> integer values corresponding to the dimensions
C
      INCLUDE 'BLOCHD1D.FOR'
      INCLUDE 'BLOCHSTK.FOR'
C
C save the value of TINC
C
      ZTINC = TINC
      ZDEL_TINC = ZDWELL
      ZT = 0.0
      NUM_PUSHES = 0
      NPHASE = ZTINC/ZDWELL
C
C reset TINC for calls to NO_RF_GRADIENTS_ON
C
      TINC = ZDWELL
C
      CALL INIT_STACK
C
      DO WHILE(ZT.LT.ZTINC)
        ZT = ZT + ZDEL_TINC
        CALL NO_RF_GRADIENTS_ON_IDEAL
C
C now get the real and imaginary data
C
        ZREAL = 0.0
        ZIMAG = 0.0
        DO Z = 1,ZDIM
              ZREAL = ZREAL + M(Z,2)
              ZIMAG = ZIMAG + M(Z,1)
        END DO
        DATA = CMPLX( ZREAL, ZIMAG )
        CALL PUSH( DATA )
        NUM_PUSHES = NUM_PUSHES + 1
      END DO
C
C adjust the stack if necessary
C
      DO WHILE(NUM_PUSHES.LE.NPHASE)
        CALL PUSH( DATA )
        NUM_PUSHES = NUM_PUSHES + 1
      END DO
      TYPE *,'NUM_PUSHES = ',NUM_PUSHES
C
C restore TINC value but don't bother with global time adjustment as
C NO_RF_GRADIENTS_ON_IDEAL takes care of it.
C
      TINC = ZTINC
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE GRADIENT( DIRECTION, START, END )
```

```
C
C This is a routine to set the gradients. It should be invoked for each
C of the three directions of gradients at each gradient step in the pulse
C program. So the gradient array, 'GRAD', should be cleared at each step
C before the new gradient values are set. This is because the gradient
C steps are added to the current gradient value!
C
C       Passed Variables:
C
C       DIRECTION       ----> a single character value indicating the
C                             direction of the gradient
C
C       START           ----> a real value for the starting value of the
C                             gradient at the edge as given in DIRECTION
C
C       END             ----> a real value for the finishing value of the
C                             gradient at the far edge as given in DIRECTION
C
C       Global Variables ( as found in BLOCHD1D.FOR ):
C
C       GRAD            ----> a three dimensional array of magnetic field
C                             gradient values. The three indices (x,y,z)
C                             define the point value of the gradient
C                             ( along the z-axis of course ) at the
C                             corresponding (x,y,z) loction in array M.
C
C       M               ----> a N+1 dimensional array of magnetization
C                             components in N space. The first N indices
C                             correspond to the coordinates of the 'spin'
C                             in space. The last index indicates the
C                             directional vector. i.e.
C                                 M(x,y,z,d) represents 'spin' x,y,z and
C                                       component d of the spin.
C                                          d = 1 is the x component
C                                           = 2 "  " y  "
C                                           = 3 "  " z  "
C
C                             The bounds on the N indices are declared
C                             in BLOCHD1D.FOR
C       .
C       .
C       .
C       XDIM
C       YDIM
C       ZDIM            ----> integer values corresponding to the dimensions
C
        INCLUDE 'BLOCHD1D.FOR'
        CHARACTER*1 DIRECTION
        REAL*4 START, END
C
C check for gradient out of bounds where the 'bounds' are as "defined"
C on the BRUKER spectrometer
C
        IF(DIRECTION.EQ.'Z'.AND.ZDIM.GT.1) THEN
          ZSTEP = (START-END)/FLOAT(ZDIM-1)
          ZTEMP = START
          DO IZ=1,ZDIM
               GRAD(IZ) = GRAD(IZ) + ZTEMP
            ZTEMP = ZTEMP - ZSTEP
          END DO
        ELSE
          TYPE *,'GRADIENT ERROR. A illegal gradient specification'
          TYPE *,'was given. Bye bye...'
          STOP
```

```
      END IF
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE PUSH( DATA )
C
C Push a complex point on top of the stack
C
      INCLUDE 'BLOCHSTK.FOR'
C
      IF(POINTER.EQ.BUFFER_LIMIT) THEN
        TYPE *,'Too much data has been acquired ! ( ILLEGAL PUSH )'
        TYPE *,'too bad, this program has expired...'
        CALL SYS$EXIT()
      ELSE
        POINTER = POINTER + 1
        ACQ_BUFFER( POINTER ) = DATA
      ENDIF
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE PULL( DATA )
C
C Pull a complex point from the top of the stack
C
      INCLUDE 'BLOCHSTK.FOR'
C
      IF(POINTER.EQ.1) THEN
        TYPE *,'Too much data has been removed from the stack !'
        TYPE *,' ( ILLEGAL PULL )'
        TYPE *,'too bad, this program has expired...'
        CALL SYS$EXIT()
      ELSE
        DATA = ACQ_BUFFER( POINTER )
        POINTER = POINTER - 1
      ENDIF
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE INIT_STACK
C
C Initialize the satck
C
      INCLUDE 'BLOCHSTK.FOR'
C
      DO I = 1, 65536
        ACQ_BUFFER(I) = ( 0.0, 0.0 )
      END DO
      BUFFER_LIMIT = 65536
C
      RETURN
      END
```

```
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE FFT1(NN)
C
C Subroutine to calculate a fast fourier transform (fft).
C This program is taken from Brighamm, E.O.: The Fast Fourier
C Transform, 164, Prentice-Hall Inc., 1974.
C
C        Passed Variables:
C
C        NN               ---> an integer value for the number of points
C                              to fourier transform. It may have values
C                              of 2**i where i = 1,2, ... ,9,10
C
      IMPLICIT INTEGER*4 (I-N)
      REAL*4 XR(1024),XI(1024),CC(1024),SS(1024)
      INTEGER*4 JP(1024,10),JBITR(1024)
      COMMON /FFTARR/ XR,XI
      DATA PI/3.14159265/
C
      IF(NN.LE.0.OR.NN.GT.1024) GO TO 200
      DUM1=FLOAT(NN)
      DUM1=ALOG(DUM1)
      DUM2=ALOG(2.0)
      DUM2=DUM1/DUM2
      NU=NINT(DUM2)
      N=2**NU
      IF(N.NE.NN) GO TO 200
      CON=2.0*PI/N
C
C SETUP SINE AND COSINE VALUES.
C
      DO 12 NU1P1=1,NU
      NU1=NU1P1-1
      DO 10 KP1=1,N
      K=KP1-1
      JP(KP1,NU1P1)=(K/2**NU1)+1
   10 CONTINUE
   12 CONTINUE
C
      DO 14 JP1=1,N
      J=JP1-1
      JBITR(JP1)=IBITR(J,NU)
      ARG=CON*FLOAT(JBITR(JP1))
      CC(JP1)=COS(ARG)
      SS(JP1)=SIN(ARG)
   14 CONTINUE
      RETURN
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
      ENTRY FFT2
      N2=N/2
      NU1=NU-1
      K=0
      DO 100 L=1,NU
  102 DO 101 I=1,N2
      KP1=K+1
      NU1P1=NU1+1
      JP1=JP(KP1,NU1P1)
      C=CC(JP1)
      S=SS(JP1)
      K1=K+1
   K1N2=K1+N2
   TREAL=XR(K1N2)*C+XI(K1N2)*S
   TIMAG=XI(K1N2)*C-XR(K1N2)*S
   XR(K1N2)=XR(K1)-TREAL
   XI(K1N2)=XI(K1)-TIMAG
   XR(K1)=XR(K1)+TREAL
   XI(K1)=XI(K1)+TIMAG
 101 K=K+1
   K=K+N2
   IF(K.LT.N) GO TO 102
   K=0
   NU1=NU1-1
 100 N2=N2/2
   DO 103 K=1,N
   I=JBITR(K)+1
   IF(I.LE.K) GO TO 103
   TREAL=XR(K)
   XR(K)=XR(I)
   XR(I)=TREAL
   TIMAG=XI(K)
   XI(K)=XI(I)
   XI(I)=TIMAG
 103 CONTINUE
   RETURN
C------------------------------------------------------------
C ERROR ROUTINE
C------------------------------------------------------------
 200 TYPE 202
 202 FORMAT('0*** FFT SUBROUTINE ERROR - ILLEGAL "N" VALUE ***')
   STOP
   END
C------------------------------------------------------------
C------------------------------------------------------------
   FUNCTION IBITR(J,NU)
   IMPLICIT INTEGER*4 (I-N)
   J1=J
   IBITR=0
   DO 200 I=1,NU
   J2=J1/2
   IBITR=IBITR*2+(J1-2*J2)
 200 J1=J2
   RETURN
   END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
	SUBROUTINE CLEAR_GRADIENTS
C
C a routine to clear the gradients. It is like RA in TOMIKON
C
	INCLUDE 'BLOCHD1D.FOR'
C
	   DO K = 1, ZDIM
	      GRAD(K) = 0.0
	   END DO
C
	RETURN
	END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
	SUBROUTINE INIT_MAGNETIZATION
```

```
C
C set initial magnetization and values for the T1 and T2 arrays
C
C        Global Variables ( as found in BLOCHD1D.FOR ):
C
C        T1A                ---> a N dimensional array containing the T1
C                               values for the corresponding magnetization
C                               vectors in array M.
C
C        T2A                ---> a N dimensional array containing the T2
C                               values for the corresponding magnetization
C                               vectors in array M.
C
C        M                  ---> a N+1 dimensional array of magnetization
C                               components in N space. The first N indices
C                               correspond to the coordinates of the 'spin'
C                               in space. The last index indicates the
C                               directional vector. i.e.
C                                   M(x,y,z,d) represents 'spin' x,y,z and
C                                         component d of the spin.
C                                             d = 1 is the x component
C                                               = 2 "  " y   "
C                                               = 3 "  " z   "
C
C                               The bounds on the N indices are declared
C                               in BLOCHD1D.FOR
C
C        .
C        .
C        .
C        XDIM
C        YDIM
C        ZDIM               ---> integer values corresponding to the dimensions
C
C
         INCLUDE 'BLOCHD1D.FOR'
C
            DO K = 1, ZDIM
                 M(K,1) = 0.0
                 M(K,2) = 0.0
                 M(K,3) = 100.0
            END DO
C
C T1 and T2 values
C
         ZT2 = 500.0E-3                           !seconds
            DO K = 1, ZDIM
                 T1A(K) = 2.0       ! seconds
                 T2A(K) = ZT2       ! seconds
            END DO
C
         RETURN
         END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
         SUBROUTINE HARD_PULSE( AXIS, TIME, ANGLE )
C
C a routine to apply a square shaped rf pulse along any axis long enough
C to tip the spins through the number of degrees given in ANGLE.
C
C        Passed Variables:
C
C        AXIS               ---> a character*2 value for the direction of H1
```

```
C                        ( can be +X, -X, +Y, -Y )
C         TIME           ---> a real value for the duration of the RF pulse
C         ANGLE          ---> a real value for the tip angle
C
C         Global Variables ( as found in BLOCHD1D.FOR ):
C
C         H1             ---> a real value for the current H1 ( or B1 ) field
C
C         GAMMA          ---> a real constant that is set to the gyromagnetic
C                             ratio for the spin species.
C
C         TINC           ---> is a real value for the incremental time step
C
          INCLUDE 'BLOCHD1D.FOR'
          REAL*8 TIME, ANGLE
          CHARACTER*2 AXIS
C
C check the axis.  If not +x, +y, -x, or -y,  then change to x
C
          IF(AXIS.NE.'+X'.AND.AXIS.NE.'-X'.AND.
          1        AXIS.NE.'+Y'.AND.AXIS.NE.'-Y') THEN
            AXIS = '+X'
          END IF
C
C calculate H1 field strength
C
          H1 = ANGLE*3.1415926/180./(TIME*GAMMA)                   ! TESLA
          ZTIME = T + TIME                                    !90 degree pulse time
          ZT = T
          TINC = TIME
          DO WHILE(TINC.GT.MAX_TINC)
            TINC = TINC/2.
          END DO
C
C do it
C
          DO WHILE(ZT.LT.ZTIME)
C
            IF(AXIS.EQ.'+X') THEN
              CALL DM_DT_PLUSX
            ELSE IF(AXIS.EQ.'-X') THEN
              CALL DM_DT_MINUSX
            ELSE IF(AXIS.EQ.'+Y') THEN
              CALL DM_DT_PLUSY
            ELSE
              CALL DM_DT_MINUSY
            END IF
C
            ZT = ZT + TINC
            IF((ZTIME-ZT).LT.TINC) TINC = (ZTIME-ZT)
          END DO
C
C turn off rf and adjust the time
C
          T = ZT
          H1 = 0.0
C
          RETURN
          END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
          SUBROUTINE HARD_90_X_IDEAL( ZTIME90 )
```

```
C
C a routine to apply a square shaped rf pulse along the x-axis long enough
C to tip the spins through 90 degrees via trigonometric calculations.
C
C       Passed Variables:
C
C       ZTIME90                  ---> a real value for the duration of the RF pulse
C
C
        INCLUDE 'BLOCHD1D.FOR'
C
        ZPI = 3.1415926
C
        DO Z = 1,ZDIM
           T1 = T1A(Z)
           T2 = T2A(Z)
C
C now find what the component in the yz plane will be in ZTIME90 seconds
C
           ZM_YZ = SQRT( M(Z,3)**2 + M(Z,2)**2 )
           IF (ABS(M(Z,2)).LE.1.E-28) THEN
               IF(M(Z,3).NE.0.0) THEN
             ZTHETA = 3.14159265 / 2.0 * M(Z,3)/ABS(M(Z,3))
               ELSE
                ZTHETA = 3.14159265 / 2.0
               END IF
           ELSE
             ZTHETA = ATAN( M(Z,3)/M(Z,2) )
             IF(M(Z,2).LT.0.0.AND.M(Z,3).LT.0.0) THEN
                ZTHETA = ZTHETA + ZPI
               END IF
             IF(M(Z,3).LT.0.0.AND.M(Z,2).GT.0.0) THEN
                ZTHETA = ZTHETA - ZPI
               END IF
           ENDIF
C
C rotate the angle 90 degrees ( from Z to Y !! )
C
              ZTHETA = ZTHETA - ZPI/2.0
C
C and reset MZ and MY ( and let MX decay a little )
C
C for Mx
C
           M(Z,1) = M(Z,1)*EXP(-ZTIME90/T2)
C
C for My
C
           M(Z,2) = ZM_YZ * COS( ZTHETA )*
        1                    EXP(-ZTIME90/T2)
C
C for Mz
C
              M(Z,3) = ZM_YZ * SIN( ZTHETA )
           M(Z,3) = M(Z,3) - (M(Z,3)-M0) *
        1                    ( 1.-EXP(-ZTIME90/T1) )
        END DO
C
        ZTIME = T + ZTIME90                         !90 degree pulse time
C
C adjust the time
C
        T = ZTIME
        H1 = 0.0
```

```
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE HARD_180_Y_IDEAL( ZTIME180 )
C
C A routine to produce a hard 180 degree pulse along the Y axis via
C trigonometric calculations.
C
C     Passed Variables:
C
C     ZTIME180          ----> a real value for the duration of the RF pulse
C
      INCLUDE 'BLOCHD1D.FOR'
C
      TINC = ZTIME180
      CALL NO_RF_GRADIENTS_ON_IDEAL        ! calculate the decays
C
C emulate a 180 along the y axis
C
      DO Z = 1,ZDIM
            M(Z,3) = M(Z,3)*-1.0
            M(Z,1) = M(Z,1)*-1.0
      END DO
C
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE SOFT_PULSE( AXIS, ZTIME, H1_VECT, NSTEPS )
C
C a routine to apply a shaped rf pulse along any axis
C long enough to tip the spins.
C
C     Passed Variables:
C
C     AXIS              ----> a character*2 value for the direction of H1
C                             ( can be +X, -X, +Y, -Y )
C     ZTIME             ----> a real value for the duration of the RF pulse
C
C     H1_VECT                    ----> a vector of 4096 real values that describe the
C                             amplitude of each point in the shaped pulse
C
C     NSTEPS            ----> an integer equal to the number of points to be
C                             used in the shaped pulse approximation
C
C     Global Variables ( as found in BLOCHD1D.FOR ):
C
C     H1                ----> a real value for the current H1 ( or B1 ) field
C
C     GAMMA             ----> a real constant that is set to the gyromagnetic
C                             ratio for the spin species.
C
C     TINC              ----> is a real value for the incremental time step
C
      INCLUDE 'BLOCHD1D.FOR'
      REAL*8 H1_VECT(4096)
      INTEGER*4 NSTEPS
      CHARACTER*2 AXIS
```

```
C
C check the axis.  If not +x, +y, -x, or -y,  then change to x
C
          IF(AXIS.NE.'+X'.AND.AXIS.NE.'-X'.AND.
          1         AXIS.NE.'+Y'.AND.AXIS.NE.'-Y') THEN
            AXIS = '+X'
          END IF
C
C set up time step size
C
          ZTIME_STEP = ZTIME/FLOAT(NSTEPS)
            ZDELTA_TIME = ZTIME_STEP
            DO WHILE(ZDELTA_TIME.GT.MAX_TINC)
              ZDELTA_TIME = ZDELTA_TIME/2.0
            END DO
          TINC = ZDELTA_TIME                                   !adjust time step
C
C do it
C
          DO ISTEP = 1,NSTEPS
            H1 = H1_VECT(ISTEP)                                !get H1 value
            TINC = ZDELTA_TIME                                 !adjust time step
            ZT = 0.0
            DO WHILE(ZT.LT.ZTIME_STEP)
C
              IF(AXIS.EQ.'+X') THEN
                CALL DM_DT_PLUSX
              ELSE IF(AXIS.EQ.'-X') THEN
                CALL DM_DT_MINUSX
              ELSE IF(AXIS.EQ.'+Y') THEN
                CALL DM_DT_PLUSY
              ELSE
                CALL DM_DT_MINUSY
              END IF
C
              ZT = ZT + TINC
              IF((ZTIME_STEP-ZT).LT.TINC) TINC = (ZTIME_STEP-ZT)
            END DO
          END DO
C
C turn off rf and adjust the time
C
          T = T + ZTIME
          H1 = 0.0
C
          RETURN
          END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
          SUBROUTINE GAUSS( ZTIME, H1_VECT, NSTEPS )
C
C  a routine ~~to place the~~ values for a gaussian shaped rf pulse into the
C  H1_VECT vector.
C
C         Parameters passed:
C
C         ZTIME            ---> a real value for the duration of the pulse
C
C         H1_VECT                     ---> a real vector of points in the shaped pulse
C
C         NSTEPS           ---> an integer value of the number of points in
C                                    H1_VECT to use to approximate the pulse
```

```
C
      INCLUDE 'BLOCHD1D.FOR'
      REAL*8 H1_VECT(4096),ENV(1024)
      INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
      ZDELTA_T = ZTIME/FLOAT(NSTEPS)                    !seconds
      ZPI = 3.1415926
      ZPI2 = ZPI/2.
      ZB1 = ZPI2/GAMMA                          !area for a hard pulse (T-s)
      ZFUDGE = 0.0                              !ZFUDGE factor start point
      ZFINC = 1.E-22                            !  "   increment
      ZSUM = 0.0
      F0 = 0.4
      ZINC = F0*NSTEPS
C
C calc envelope
C
      DO I = 1, NSTEPS
        X0 = (2*I-NSTEPS+2)/ZINC
        ENV(I) = DFLOAT(INT(255.*EXP(-1.0*ZPI*X0**2)))
      END DO
C
      DO WHILE(ABS(1.-ABS(ZSUM/ZB1)).GE.1.0E-7)
        IF(ZSUM.LT.ZB1) THEN
          IF(ZSUM.NE.0.0) ZFINC = ZFINC + ZFINC*.5*(ZB1/ZSUM)
          ZFUDGE = ZFUDGE + ZFINC
        ELSE
          IF(ZSUM.NE.0.0) ZFINC = ZFINC - ZFINC*.5*(ZB1/ZSUM)
          ZFUDGE = ZFUDGE - ZFINC
        END IF
        ZGFACT = ZFUDGE*ZDELTA_T
        ZSUM = 0.0
        DO I = 1, NSTEPS
          Y0 = ENV(I)*ZGFACT
          ZSUM = ZSUM + Y0
        END DO
      END DO
      ZGFACT = ZGFACT/ZDELTA_T
C
C Stuff it all into the envelope vector
C
      DO I = 1,NSTEPS
        H1_VECT(I) = ENV(I)*ZGFACT
      END DO
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE GSINC( ZTIME, H1_VECT, NSTEPS )
C
C  a routine to place the values for a gaussian enveloped, sinc shaped
C  rf pulse into the H1_VECT vector.
C
C     Parameters passed:
C
C     ZTIME              ---> a real value for the duration of the pulse
C
C     H1_VECT                   ---> a real vector of points in the shaped pulse
C
C     NSTEPS             ---> an integer value of the number of points in
```

```
C                     H1_VECT to use to approximate the pulse
C
      INCLUDE 'BLOCHD1D.FOR'
      REAL*8 H1_VECT(4096)
      INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
C     ZTIME = FLOAT(NSTEPS)*7.5E-6              !seconds
      ZTWOPI = 2.*3.1415926
      ZPI2 = 3.1415926/2.
      ZD = 95./64.
      Z2D2 = 2.*ZD**2
      ZB1_TW = ZPI2/GAMMA                       !area for a hard pulse (T-s)
      FUDGE = 1.2                               !fudge factor start point
      FINC = 5.E-6                              !  "   increment
      H = 8.0/(FLOAT(NSTEPS))
      ZH = ZTIME/FLOAT(NSTEPS)
      SUM = 0.0
      DO WHILE(ABS(1.-ABS(SUM/ZB1_TW)).GE.1.0E-5)
        X0 = -4.0
        IF(SUM.LT.ZB1_TW) THEN
          FINC = FINC*2.0
          OLDFUDGE = FUDGE
          FUDGE = FUDGE + FINC
        ELSE
          FINC = FINC*0.499998
          FUDGE = (FUDGE + OLDFUDGE)/2.0
        END IF
        GFACTOR = ZB1_TW*FUDGE
        SUM = 0.0
        DO I = 1, NSTEPS
          IF(X0.EQ.0.0) THEN
            ZSINC = 1.0
          ELSE
            ZSINC = SIN(ZTWOPI*X0)/(ZTWOPI*X0)
          END IF
          Y0 = EXP(-1.0*X0**2/Z2D2)*ZH*GFACTOR*ZSINC
          X0 = X0 + H
          SUM = SUM + Y0
        END DO
      END DO
C
      X1 = -4.0
      X1INC = 8.0/FLOAT(NSTEPS)
      ZSUM = 0.0
      DO I = 1,NSTEPS
          IF(X1.EQ.0.0) THEN
            ZSINC = 1.0
          ELSE
            ZSINC = SIN(ZTWOPI*X1)/(ZTWOPI*X1)
          END IF
        H1_VECT(I) = EXP(-1.0*X1**2/Z2D2)*GFACTOR*ZSINC
        X1 = X1 + X1INC
      END DO
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE HSINC( ZTIME, H1_VECT, NSTEPS )
C
C  a routine to place the values for a Hanning enveloped, sinc shaped
```

```
C  rf pulse into the H1_VECT vector.
C
C        Parameters passed:
C
C        ZTIME          ----> a real value for the duration of the pulse
C
C        H1_VECT                 ----> a real vector of points in the shaped pulse
C
C        NSTEPS         ----> an integer value of the number of points in
C                             H1_VECT to use to approximate the pulse
C
         INCLUDE 'BLOCHD1D.FOR'
         REAL*8 H1_VECT(4096),ENV(1024)
         INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
C        ZTIME = FLOAT(NSTEPS)*7.5E-6           !seconds
         ZPI = 3.1415926
         ZTWOPI = 2.*ZPI
         ZPI2 = ZPI/2.
         ZD = 95./64.
         Z2D2 = 2.*ZD**2
         ZB1_TW = ZPI2/GAMMA                    !area for a hard pulse (T-s)
         FUDGE = 1.2                            !fudge factor start point
         FINC = 5.E-6                           ! "   increment
         H = 8.0/(FLOAT(NSTEPS))
         ZH = ZTIME/FLOAT(NSTEPS)
C
C calc envelope
C
          X0 = -4.0
          DO I = 1, NSTEPS
            IF(X0.EQ.0.0) THEN
              ZSINC = 1.0
            ELSE
              ZSINC = SIN(ZTWOPI*X0)/(ZTWOPI*X0)
            END IF
            ZWINDOW = .5*(1+COS(ZPI*X0/4.0))
            ENV(I) = ZWINDOW*ZSINC
            X0 = X0 + H
          END DO
C
         SUM = 0.0
         DO WHILE(ABS(1.-ABS(SUM/ZB1_TW)).GE.1.0E-5)
          IF(SUM.LT.ZB1_TW) THEN
            FINC = FINC*2.0
            OLDFUDGE = FUDGE
            FUDGE = FUDGE + FINC
          ELSE
            FINC = FINC*0.499998
            FUDGE = (FUDGE + OLDFUDGE)/2.0
          END IF
          GFACTOR = ZB1_TW*FUDGE
          SUM = 0.0
          DO I = 1, NSTEPS
            Y0 = ENV(I)*ZH*GFACTOR
            SUM = SUM + Y0
          END DO
         END DO
C
         DO I = 1,NSTEPS
          H1_VECT(I) = ENV(I)*GFACTOR
         END DO
```

```
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE GAUSS180( ZTIME, H1_VECT, NSTEPS )
C
C a routine to place the values for a gaussian shaped rf pulse into the
C H1_VECT vector.
C
C     Parameters passed:
C
C     ZTIME           ----> a real value for the duration of the pulse
C
C     H1_VECT                 ----> a real vector of points in the shaped pulse
C
C     NSTEPS          ----> an integer value of the number of points in
C                             H1_VECT to use to approximate the pulse
C
      INCLUDE 'BLOCHD1D.FOR'
      REAL*8 H1_VECT(4096),ENV(1024)
      INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
      ZDELTA_T = ZTIME/FLOAT(NSTEPS)                !seconds
      ZPI = 3.1415926
      ZPI2 = ZPI/2.
      ZB1 = ZPI/GAMMA                         !area for a hard pulse (T-s)
      ZFUDGE = 0.0                            !ZFUDGE factor start point
      ZFINC = 1.E-22                          ! "  increment
      ZSUM = 0.0
      F0 = 0.4
      ZINC = F0*NSTEPS
C
C calc envelope
C
      DO I = 1, NSTEPS
        X0 = (2*I-NSTEPS+2)/ZINC
        ENV(I) = DFLOAT(INT(255.*EXP(-1.0*ZPI*X0**2)))
      END DO
C
      DO WHILE(ABS(1.-ABS(ZSUM/ZB1)).GE.1.0E-7)
        IF(ZSUM.LT.ZB1) THEN
          IF(ZSUM.NE.0.0) ZFINC = ZFINC + ZFINC*.5*(ZB1/ZSUM)
          ZFUDGE = ZFUDGE + ZFINC
        ELSE
          IF(ZSUM.NE.0.0) ZFINC = ZFINC - ZFINC*.5*(ZB1/ZSUM)
          ZFUDGE = ZFUDGE - ZFINC
        END IF
        ZGFACT = ZFUDGE*ZDELTA_T
        ZSUM = 0.0
        DO I = 1, NSTEPS
          Y0 = ENV(I)*ZGFACT
          ZSUM = ZSUM + Y0
        END DO
      END DO
      ZGFACT = ZGFACT/ZDELTA_T
C
C
      DO I = 1,NSTEPS
        H1_VECT(I) = ENV(I)*ZGFACT
      END DO
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      SUBROUTINE GSINC180( ZTIME, H1_VECT, NSTEPS )
C
C a routine to place the values for a gaussian enveloped, sinc shaped
C rf pulse into the H1_VECT vector.
C
C       Parameters passed:
C
C       ZTIME           ----> a real value for the duration of the pulse
C
C       H1_VECT                 ----> a real vector of points in the shaped pulse
C
C       NSTEPS          ----> an integer value of the number of points in
C                             H1_VECT to use to approximate the pulse
C
      INCLUDE 'BLOCHD1D.FOR'
      REAL*8 H1_VECT(4096),ENV(1024)
      INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
C     ZTIME = FLOAT(NSTEPS)*7.5E-6            !seconds
      ZTWOPI = 2.*3.1415926
      ZPI2 = 3.1415926/2.
      ZD = 95./64.
      Z2D2 = 2.*ZD**2
      ZB1_TW = ZPI2*2./GAMMA                          !area for a hard pulse (T-s)
      ZFUDGE = 0.0                            !fudge factor start point
      ZFINC = 5.E-22                          ! "  increment
      H = 8.0/(FLOAT(NSTEPS))
      ZH = ZTIME/FLOAT(NSTEPS)
C
C calc envelope
C
      X0 = -4.0
      DO I = 1, NSTEPS
        IF(X0.EQ.0.0) THEN
          ZSINC = 1.0
        ELSE
          ZSINC = SIN(ZTWOPI*X0)/(ZTWOPI*X0)
        END IF
        ENV(I) = EXP(-1.0*X0**2/Z2D2)*ZSINC
        X0 = X0 + H
      END DO
C
      ZSUM = 0.0
      DO WHILE(ABS(1.-ABS(ZSUM/ZB1_TW)).GE.1.0E-5)
        IF(ZSUM.LT.ZB1_TW) THEN
          IF(ZSUM.NE.0.0) ZFINC = ZFINC + ZFINC*.5*(ZB1_TW/ZSUM)
          ZFUDGE = ZFUDGE + ZFINC
        ELSE
          IF(ZSUM.NE.0.0) ZFINC = ZFINC - ZFINC*.5*(ZB1_TW/ZSUM)
          ZFUDGE = ZFUDGE - ZFINC
        END IF
        GFACTOR = ZB1_TW*ZFUDGE
        ZSUM = 0.0
        DO I = 1, NSTEPS
          Y0 = ENV(I)*ZH*GFACTOR
          ZSUM = ZSUM + Y0
          END DO
        END DO
C
        DO I = 1,NSTEPS
          H1_VECT(I) = ENV(I)*GFACTOR
        END DO
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
        SUBROUTINE HSINC180( ZTIME, H1_VECT, NSTEPS )
C
C  a routine to place the values for a Hanning enveloped, sinc shaped
C  rf pulse into the H1_VECT vector.
C
C       Parameters passed:
C
C       ZTIME             ----> a real value for the duration of the pulse
C
C       H1_VECT                   ----> a real vector of points in the shaped pulse
C
C       NSTEPS            ----> an integer value of the number of points in
C                               H1_VECT to use to approximate the pulse
C
        INCLUDE 'BLOCHD1D.FOR'
        REAL*8 H1_VECT(4096),ENV(1024)
        INTEGER*4 NSTEPS
C
C find the area ( crudely ) under the curve for the number of steps passed
C
C       ZTIME = FLOAT(NSTEPS)*7.5E-6              !seconds
        ZPI = 3.1415926
        ZTWOPI = 2.*ZPI
        ZPI2 = ZPI/2.
        ZD = 95./64.
        Z2D2 = 2.*ZD**2
        ZB1_TW = ZPI/GAMMA                        !area for a hard pulse (T-s)
        ZFUDGE = 0.0                              !fudge factor start point
        ZFINC = 5.E-22                            ! "  increment
        H = 8.0/(FLOAT(NSTEPS))
        ZH = ZTIME/FLOAT(NSTEPS)
C
C calc envelope
C
          X0 = -4.0
          DO I = 1, NSTEPS
            IF(X0.EQ.0.0) THEN
              ZSINC = 1.0
            ELSE
              ZSINC = SIN(ZTWOPI*X0)/(ZTWOPI*X0)
            END IF
            ZWINDOW = .5*(1+COS(ZPI*X0/4.0))
            ENV(I) = ZWINDOW*ZSINC
            X0 = X0 + H
          END DO
C
        ZSUM = 0.0
        DO WHILE(ABS(1.-ABS(ZSUM/ZB1_TW)).GE.1.0E-5)
          IF(ZSUM.LT.ZB1_TW) THEN
            IF(ZSUM.NE.0.0) ZFINC = ZFINC + ZFINC*.5*(ZB1_TW/ZSUM)
            ZFUDGE = ZFUDGE + ZFINC
          ELSE
          IF(ZSUM.NE.0.0) ZFINC = ZFINC - ZFINC*.5*(ZB1_TW/ZSUM)
          ZFUDGE = ZFUDGE - ZFINC
        END IF
        GFACTOR = ZB1_TW*ZFUDGE
        ZSUM = 0.0
        DO I = 1, NSTEPS
          Y0 = ENV(I)*ZH*GFACTOR
          ZSUM = ZSUM + Y0
        END DO
      END DO
C
      DO I = 1,NSTEPS
        H1_VECT(I) = ENV(I)*GFACTOR
      END DO
      RETURN
      END
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCC
```

The INCLUDE files.

The first include file, BLOCHDID.FOR, is used throughout the simulation package. The second file, BLOCHSTK.FOR, is used in the data sampling routines.

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
ccccccc
C
C file of declarations for the BLOCH Equation simulation program
C
        IMPLICIT REAL*8(Z)
        REAL*8 W, GAMMA, M(20,3), H1, OMEGA, T1, T2, T, M0,
        1    GRAD(20), TINC, MAX_TINC, T1A(20),
        2    T2A(20)
C
        INTEGER*4 N, X, Y, Z, XDIM, YDIM, ZDIM, NPHASE, NFREQ
C
        COMMON/PARM/ N, GAMMA, M, H1, OMEGA, T1, T2, T, M0,
        1             X, Y, Z, GRAD, TINC, NPHASE, NFREQ, MAX_TINC,
        2             T1A,T2A
C
        DATA ZDIM/20/    ! array dimensions
        DATA MAX_TINC/100.E-6/, N/128/  !maximum time step value,
                                        !number of Runge kutta steps


cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccc
C
C Define stack variables
C
        COMPLEX*8 ACQ_BUFFER(65536), DATA
        INTEGER*4 POINTER, BUFFER_LIMIT
C
        COMMON /ACQ/ ACQ_BUFFER, POINTER, BUFFER_LIMIT
```

A Sample pulse program.

This program is based on some SPARS (spatially resolved spectroscopy) experiments that Dr. Chris Hanstock has been working on.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCC
C
      SUBROUTINE PLOT( FILENAME2 )
C
C A routine to place the spin data into plot files for my plotting routine
C to eat.
C
C     Plot file format:
C     ----------------
C
C     Block #1
C     ++++++++
C
C     CHARACTER*80        x axis label string
C     CHARACTER*80        y axis label string
C     REAL*4              xmax
C     REAL*4              xmin
C     REAL*4              ymax
C     REAL*4              ymin
C     CHARACTER*80        title of plot
C     INTEGER*4           size of plot; i.e. the number of x y pairs
C
C     Block #2
C     ++++++++
C
C     INTEGER*2           x1
C     INTEGER*2           y1
C     INTEGER*2           x2
C     INTEGER*2           y2
C     INTEGER*2           x3
C     INTEGER*2           y3
C     INTEGER*2           x4
C
C                   and so on...
C
      INCLUDE 'BLOCHD1D.FOR'
      INTEGER*2 BUFFER(65536), READBUF(256)
      REAL*4 XMAX,XMIN,YMAX,YMIN
      CHARACTER*80 XLABEL,YLABEL,TITLE
C
      CHARACTER*40 FILENAME, FILENAME2
      BYTE FILE(40)
      EQUIVALENCE( FILE(1), FILENAME )
C
C Twiddle filenames
C
      FILENAME = FILENAME2
      I = 40
      DO WHILE(I.NE.1)
        IF(FILE(I).NE.' ') THEN
          ILEN = I
          I = 2
          END IF
          I = I - 1
        END DO
        FILE(ILEN+1) = '.'
        FILE(ILEN+2) = 'W'
        DO IFILES = 1,4
          OPEN( UNIT=2, FILE=FILENAME, TYPE='NEW',
     1        FORM='UNFORMATTED',ERR=999)
C
C
          NBLOCKS = ZDIM/256 + 1
          J = 1
          DO I = 1, ZDIM*2, 2
            BUFFER(I) = J
            J = J + 1
          END DO
C
C convert to I*4 format
C
          IF(IFILES.EQ.1) THEN               ! find component in xy plane
            J = 1
            DO I = 2, ZDIM*2, 2
                  BUFFER(I) = INT( SQRT( M(J,1)**2 + M(J,2)**2 ) )
              J = J + 1
            END DO
          ELSE
            J = 1
            DO I = 2, ZDIM*2, 2
              BUFFER(I) = INT( M(J,IFILES-1 ))
              J = J + 1
            END DO
          END IF
C
C save the number of points and write out to the plot file
C
          WRITE(2) XLABEL,YLABEL,XMAX,XMIN,YMAX,YMIN,TITLE,ZDIM
          DO I = 1, ZDIM/128 + 1
            DO J = 1, 256
              K = ( I - 1 )*256 + J
              READBUF(J) = BUFFER(K)
            END DO
            WRITE(2) READBUF
          END DO
            CLOSE(2)
            FILE(ILEN+2) = FILE(ILEN+2)+1
        END DO
C
        RETURN
 999    TYPE *,'*** Error opening plot file'
        STOP
 1000   FORMAT(A)
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCC
C
        SUBROUTINE PLOT_ACQ_DAT( FILENAME2, INBUF, NUMPOINTS )
C
C A routine to place acquired data into plot files for my plotting routine
C to eat.
C
C       plot file format:
C       ---------------
C
```

```
C       Block #1
C       ++++++++
C
C       CHARACTER*80            x axis label string
C       CHARACTER*80            y axis label string
C       REAL*4                  xmax
C       REAL*4                  xmin
C       REAL*4                  ymax
C       REAL*4                  ymin
C       CHARACTER*80            title of plot
C       INTEGER*4               size of plot; i.e. the number of x y pairs
C
C       Block #2
C       ++++++++
C
C       INTEGER*2               x1
C       INTEGER*2               y1
C       INTEGER*2               x2
C       INTEGER*2               y2
C       INTEGER*2               x3
C       INTEGER*2               y3
C       INTEGER*2               x4
C
C                          and so on...
C
C       Parameters passed
C       +++++++++++++++++
C
C       INBUF        ---> real*4 vector of Y data points
C
C       FILENAME2    ---> character*40 file name
C
C       NUMPOINTS    ---> integer*4 giving the number of plot points
C
C
        INCLUDE 'BLOCHD1D.FOR'
        INTEGER*4 NUMPOINTS
        INTEGER*2 OUTBUF(65536), READBUF(256)
        REAL*4 XMAX,XMIN,YMAX,YMIN, INBUF(65536)
        CHARACTER*80 XLABEL,YLABEL,TITLE
C
        CHARACTER*40 FILENAME, FILENAME2
        BYTE FILE(40)
        EQUIVALENCE( FILE(1), FILENAME )
C
C Twiddle filenames
C
        FILENAME = FILENAME2
        I = 40
        DO WHILE(I.NE.1)
          IF(FILE(I).NE.' ') THEN
            ILEN = I
            I = 2
          END IF
          I = I - 1
        END DO
        FILE(ILEN+1) = '.'
        FILE(ILEN+2) = 'P'
        FILE(ILEN+3) = 'L'
        FILE(ILEN+4) = 'T'
        OPEN( UNIT=2, FILE=FILENAME, TYPE='NEW',
        1     FORM='UNFORMATTED',ERR=999)
C
C
```

```
      NBLOCKS = NUMPOINTS/256 + 1
      J = 1
      DO I = 1, NUMPOINTS*2, 2
        OUTBUF(I) = J
        J = J + 1
      END DO
C
C scale and stuff INBUF stuff into OUTBUF at alternate locations
C
      ZMAX = -1.E30
      ZMIN = 1.E30
      DO I = 1,NUMPOINTS
        IF(ZMIN.GT.INBUF(I)) ZMIN = INBUF(I)
        IF(ZMAX.LT.INBUF(I)) ZMAX = INBUF(I)
      END DO
      IF((ZMAX-ZMIN).NE.0.0) THEN
        IF((ZMAX-ZMIN).GT.ZMAX) THEN
          ZSCALE = 32765./(ZMAX-ZMIN)
        ELSE
          ZSCALE = 32765./ZMAX
        END IF
      ELSE
        ZSCALE = 1.
      END IF
C
      DO I = 1,NUMPOINTS
        J = I * 2
        OUTBUF(J) = INT(ZSCALE*INBUF(I))
      END DO
C
C save the number of points and write out to the plot file
C
      WRITE(2) XLABEL,YLABEL,XMAX,XMIN,YMAX,YMIN,TITLE,NUMPOINTS
      DO I = 1, NUMPOINTS/128 + 1
        DO J = 1, 256
          K = ( I - 1 )*256 + J
          READBUF(J) = OUTBUF(K)
        END DO
        WRITE(2) READBUF
      END DO
      CLOSE(2)
C
      RETURN
999   TYPE *,'**** Error opening plot file'
      STOP
1000  FORMAT(A)
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCC
C
      PROGRAM SPARS
C
C A program to investigate the behavior of the magnetizations during part
C of CCH's SPARS ( SPAtially Resolved Spectroscopy ) routine.
C
      INCLUDE 'BLOCHD1D.FOR'
      INCLUDE 'BLOCHSTK.FOR'
      INTEGER*4 NUMPOINTS
      INTEGER*4 READBUF(128),NSTEPS
      REAL*8 H1_VECT(4096)
      REAL*4 XR(1024),XI(1024), BUFFER(65536)
```

```
C
      CHARACTER*40 FILENAME
      BYTE FILE(40)
      EQUIVALENCE( FILE(1), FILENAME )
      COMMON/FFTARR/ XR,XI
C
C initialize the volume
C
      CALL INIT_MAGNETIZATION
      CALL CLEAR_GRADIENTS
C
C set constants
C
      ZPI = 3.1415926
      OMEGA = 0.0
      GAMMA = 2*ZPI*100.0E6 / 2.35      ! Rads/ sec / TESLA for protons
      M0 = 100.0                        ! ?
      T = 0.0
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCC
C
C first put all the spins in the x y plane
C
      CALL HARD_PULSE( '+X', 125.E-6, 90. )
C
C then dephase the spins using a z gradient
C
      CALL CLEAR_GRADIENTS
      ZGZ = 118.815E-6/2.
      CALL GRADIENT( 'Z', ZGZ, -1.*ZGZ )
      TINC = 5.84E-3
      CALL NO_RF_GRADIENTS_ON_IDEAL
      CALL CLEAR_GRADIENTS
C
C reverse the spins
C
      CALL HARD_PULSE( '+Y', 250.E-6, 180. )
C
C and start to rephase the spins
C
      CALL CLEAR_GRADIENTS
      ZGZ = 118.815E-6/2.
      CALL GRADIENT( 'Z', ZGZ, -1.*ZGZ )
      TINC = 2.E-3
      CALL NO_RF_GRADIENTS_ON_IDEAL
C
C now apply a soft 90 to select the slice of interest.  Note that the
C rest of the spins outside the slice start to dephase midway through
C the soft pulse and continue until they are completely dephased
C
      NSTEPS = 1024
      ZTIME = 7.68E-3
      CALL GAUSS( ZTIME, H1_VECT, NSTEPS )
      CALL SOFT_PULSE( '+X', ZTIME, H1_VECT, NSTEPS )
C
C here one would in principle wait a while,  but we want to see a spin
C echo soon so,...
C
      CALL CLEAR GRADIENTS
      CALL HARD_PULSE( '+X', 125.E-6, 90. )
      ZGZ = 118.815E-6/2.
      CALL GRADIENT( 'Z', ZGZ, -1.*ZGZ )
      TINC = 6.4E-3
      CALL NO_RF_GRADIENTS_ON_IDEAL
      CALL CLEAR_GRADIENTS
C
C 180 along plus y axis
C
      CALL HARD_PULSE( '+Y', 250.E-6, 180. )
C
C set up refocussing gradient and acquire a spin echo
C
C the acquisition data structure must be initialized first
C
      CALL INIT_STACK
C
      ZGZ = 118.815E-6/2.
      CALL GRADIENT( 'Z', ZGZ, -1.*ZGZ )
      TINC = 12.8E-3
      ZDWELL = 25.E-6
      CALL ACQUIRE_SIGNAL( ZDWELL )
C
C do the FT right now so that everything can be output later on.
C
      DO KS = 1, 512                        ! THE NUMBER OF SAMPLE POINTS
        XR(KS) = REAL( ACQ_BUFFER(KS) )
        XI(KS) = AIMAG( ACQ_BUFFER(KS) )
      END DO
      CALL FFT1( 512 )
      CALL FFT2
C
C scale the frequency domain results
C
      DO I = 1,512
        XR(I) = XR(I)/512
        XI(I) = XI(I)/512
      END DO
C
C an flip about the centre axis
C
      DO I = 1,128
        J = 257 - I
        ZRTEMP = XR(I)
        XR(I) = XR(J)
        XR(J) = ZRTEMP
        ZITEMP = XI(I)
        XI(I) = XI(J)
        XI(J) = ZITEMP
      END DO
      DO I = 257,384
        J = 513 - I + 256
        ZRTEMP = XR(I)
        XR(I) = XR(J)
        XR(J) = ZRTEMP
        ZITEMP = XI(I)
        XI(I) = XI(J)
        XI(J) = ZITEMP
      END DO
C
C now the FFTed results are in XI and XR
C
C now place time domain data into the BUFFER and plot it
C
C The real time data
C
      DO I = 1, 512
        BUFFER(I) = REAL(ACQ_BUFFER(I))
      END DO
```

```
        FILENAME = 'SPARSRT'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
C The imaginary time data
C
        DO I = 1, 512
          BUFFER(I) = AIMAG(ACQ_BUFFER(I))
        END DO
        FILENAME = 'SPARSIT'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
C The modulus time data
C
        DO I = 1, 512
          BUFFER(I) = CABS(ACQ_BUFFER(I))
        END DO
        FILENAME = 'SPARSMT'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
C save the frequency domain data in plot files
C
C The real freq data
C
        DO I = 1, 512
          BUFFER(I) = XR(I)
        END DO
        FILENAME = 'SPARSRF'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
C The imaginary freq data
C
        DO I = 1, 512
          BUFFER(I) = XI(I)
        END DO
        FILENAME = 'SPARSIF'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
C The modulus freq data
C
        DO I = 1, 512
          BUFFER(I) = SQRT(XI(I)*XI(I) + XR(I)*XR(I))
        END DO
        FILENAME = 'SPARSMF'
        CALL PLOT_ACQ_DAT( FILENAME, BUFFER, 512 )
C
        STOP
C
 999    TYPE *,'**** Error opening plot file'
        STOP
C
 234    FORMAT(' X ',G14.7,' Y ',G14.7,' Z ',G14.7)
 900    FORMAT('0 THE COMPLEX DATA POINTS ',/,/)
 909    FORMAT(' ',I4,7X,G14.7,4X,'I',G14.7,/)
 919    FORMAT('0 THE FFTed RESULTS')
 1000   FORMAT(A)
        END
```

Appendix E

This is a listing of the program used to check the results of the experiment simulating a square 90 RF pulse with a gradient along the z axis. The equations were taken from a paper on selective RF pulse simulations (Locher, 1980).

```
cccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
cccccccccccc
C
C A quick and dirty program to calculate the magnetization components after the application of a
C square 90 degree pulse under a gradient. The equations are given in Locher's paper of 1980.
C
      IMPLICIT REAL*8(Z)
      REAL*8 H0, H1, TAU, GAMMA, THETA, OMEGA, F, G,
      1     MX, MY, MZ, GZ_LT, GZ_RT, PHI_MXY
      INTEGER*4 NUM_PTS
      CHARACTER*1 ANS1, ANS2
C
C do some calculations from Locher's paper ( Phil. Trans. R. Soc. Lond.
C B 289, 537-542 (1980) )
C
      H0 = 2.35
      ZPI = 3.1415926
      GAMMA = 2.* ZPI * 100.E6 / 2.35
      ANS1 = 'Y'
      DO WHILE(ANS1.EQ.'Y')
C
C get data from user
C
        WRITE(6,500)
        ACCEPT *,THETA
        THETA = THETA*ZPI/180.
        WRITE(6,501)
        ACCEPT *,TAU
        H1 = ZPI/2./GAMMA/TAU
C
        WRITE(6,502)
        ACCEPT *,NUM_PTS
        WRITE(6,503)
        ACCEPT *,GZ_LT,GZ_RT
        ZSLOPE = (GZ_RT - GZ_LT)/(NUM_PTS-1)
        ZINTERCEPT = GZ_LT - ZSLOPE
        ANS2 = 'Y'
        DO WHILE(ANS2.EQ.'Y')
          WRITE(6,504)
          ACCEPT *,IPT
          ZGZ = ZINTERCEPT + ZSLOPE*FLOAT(IPT)
          F = -1.*ZGZ/H1
          G = SQRT( 1. + F*F )
          MY = 100.*SIN( THETA*G )/G
          MX = 100.*F*( 1 - COS( THETA*G ) )/(G*G)
          MZ = 100.*( F*F + COS( THETA*G ) )/(G*G)
          ZMY = MY
          ZMX = -1.*MX
           IF(MY.LE.1.E-20.AND.MY.GE.-1.E-20) ZMY = 1.E-20
          PHI_MXY = ATAN(ZMX/ZMY)
           IF(ZMY.LT.0.0.AND.ZMX.LT.0.0) PHI_MXY = PHI_MXY - ZPI
           IF(ZMY.LT.0.0.AND.ZMX.GT.0.0) PHI_MXY = PHI_MXY + ZPI
          PHI_MXY = PHI_MXY * 180 / ZPI
          WRITE(6,505) IPT, MX, MY, MZ, PHI_MXY
          ANS2 = ''
           DO WHILE(ANS2.NE.'Y'.AND.ANS2.NE.'N')
                 WRITE(6,506)
                 READ(5,1000) ANS2
          END DO
         END DO
         ANS1 = ''
         DO WHILE(ANS1.NE.'Y'.AND.ANS1.NE.'N')
           WRITE(6,507)
           READ(5,1000) ANS1
         END DO
       END DO
       STOP
500    FORMAT(/,' Enter a tip angle in degrees : ',$)
501    FORMAT(/,' Enter the time the RF pulse is applied in secs : ',$)
502    FORMAT(/,' Enter the number of points along the z direction : ',$)
503    FORMAT(/,' Enter Gz value at point 1 and Gz value at point N : ',$)
504    FORMAT(/,' Enter the point of interest : ',$)
505    FORMAT(/,' POINT ',I3,' MX = ',G14.7,' MY = ',G14.7,' MZ = ',G14.7,
       1          ' PHI = ',G14.7)
506    FORMAT(/,' Another point ( Y or N ) ? : ',$)
507    FORMAT(/,' Another set of parameters ( Y or N ) ? : ',$)
C
1000   FORMAT(A)
       END
```

Appendix F

The command file IMFL (IMaging of FLow).

```
------ FILE:  IMFL

;  DATE OF LAST EDITTING:  5.10.86
;
1@DF@RF2@DF@RF3@DF@RF
1STRT,-1,2STRT,-1,3STRT,-1,
1RMEM,20,2RMEM,34,3RMEM,4
1RJ FTHP,AI,0,FL FQLIST,@CTL,RES SHIMVAL,RA
2RJ IMGEFLJ,AI,0,FL FQLIST,@CTL,RES SHIMVAL,RA
3RJ SETUP,AI,0,FL FQLIST,@CTL,RES SHIMVAL,RA
3@DN2@DN1@DN
1RE FTHP.001,AU IMGEO1,3AU IMGEO1
2RE IMFL.001,REP,,AU IMGEO1
RPF,QPF,,AP,,
DW,24,AQ,
@P
START ONE OF THE FOLLOWING EXECUTIONS:

IMFLPL            FLOW BY PHASE            COMP.:  GRAD
@
```

Appendix G

The command file IMFLPL (IMaging of FLow, pulse Program Load). The pulse program PHAS.PC, the gradient generation file IMFLG.EXE, and the pulse program execution file IMGEGOO are also included.

```
------ FILE:  IMFLPL

; DATE OF LAST EDITTING: 5.10.86
;
@DF
2
SW,20833,AQ,,
RP PHAS,D11,.8S,C2,1
;
; D11 = TR
; C2 = NUMBER OF ECHOES
NI,1,NA,1,BSI,512WAQ,,
FL FQLIST,@CTLAU IMGEO1
;
; ADJUST SOFT PULSE
;
2REG SLZ
GZ,M,2,GPHSZM.021
@P
              CHOOSE A GRADIENT STRENGTH

              GPHSZM.021     ==>     IOD = 500
              GPHSZM.022     ==>     IOD = 220
              GPHSZM.023     ==>     IOD = 147
              GPHSZM.024     ==>     IOD = 100
@
GZ,,,@G
@P
ADJUST CHANNEL 1 (ESC)
@
2AZ,NE,1000,SI,512WAQ,,
@DN@RN@CTY
AU IMGEGOO,@W@S
@CTY
REG IMGEMTR,@DF
;
; SET GRADIENTS UP
;
GX,,GPHMERXC.021,GY,,GPHMEPYC.121,GZ,,GPHMESZC.021
@P
START ONE OF THE FOLLOWING EXECUTIONS
AFTER READING IN THE PROPER GRADIENTS
AND WRG IMGEMTR :

       GX  GPHMERXC.02%     GY  GPHMEPYC.*2%     GZ  GPHMESZC.02#

       % = 1  0 ZOOM        * = 1   128 GRAD. STEPS      # = 1  IOD = 500
           2 1  "                                            2  IOD = 220
                                                             3  IOD = 147
                                                             4  IOD = 90
```

5 IOD = 54

```
NEXT, START EXECUTION IMFLG
@


------ FILE: IMGEGOO

O1EN
O1
STRT 0
SI 512W
VDS 9
1               GOD
                DISP
IN 1
STPP
EXIT


------ FILE: PHAS.PC

; SINGLE SLICE   MULTI ECHO   FLOW BY PHASE
;
;         C2 = NUMBER OF ECHOES
;         D11 = TR ( REPETITION TIME )

START,
;                 **** ADD SWEEP
;
;
          20M      [XT+X]  <CNG>                       ;SLICE GRAD
          LOOP 1024 TIMES                              ;SHAPED PULSE
           3.75U    [F1+X XT-X XT+Y XAD]               ; (+X)
           3.75U    [F1+X XT-X XT+Y]
          END LOOP
          3M       [XT+X]  <CNG>                       ;FREQUENCY ENCODING
          1M                                           ;DEPHASING GRAD ON
          2M       [XT+X]  <CNG>                       ;BALANCED FLOW COMP ON
          2M
          1304U    [XT+X]  <CNG>                       ;FREQ AND COMP GRAD OFF
          1M
          LOOP C2 TIMES
           3M                                          ;ALL GRADS OFF
           250U     [F1-Y]                             ;180 (-Y)
           2M       [XT+X]  <CNG>                      ;Z GRAD REFOCUSSING
           2M                                          ;AND PHASE ENCODING ON
           3M       [XT+X]  <CNG>                      ;ALL GRADS OFF
           1M
           3M       [XT+X]  <CNG TCHK>                 ;READOUT ON, TIMING CHECK
           AQ       [STA]                              ;START ACQUISITION
           2M       [XT+X]  <CNG>                      ;FREQ COMP FOR NEXT ECHO
           1M
           3M       [XT+X]  <CNG>                      ;ALL GRADS OFF
           2M
          END LOOP
          1M       [XT+X]  <CNG>                       ;STOP POINT
          D11                                          ;TR
;
;                 **** SUBTRACT SWEEP
;
          20M      [XT+X]  <CNG>                       ;SLICE GRAD
          LOOP 1024 TIMES                              ;SHAPED PULSE
           3.75U    [F1+X XT-X XT+Y XAD]               ; (+X)
           3.75U    [F1+X XT-X XT+Y]
```

```
        END LOOP
        3M       [XT+X]  <CNG>                         ;FREQUENCY ENCODING
        1M                                             ;DEPHASING GRAD ON
        2M       [XT+X]  <CNG>                         ;BALANCED FLOW COMP ON
        2M
        1304U    [XT+X]  <CNG>                         ;FREQ AND COMP GRAD OFF
        1M
        LOOP C2 TIMES
         3M                                            ;ALL GRADS OFF
         250U      [F1-Y]                              ;180 (-Y)
         2M        [XT+X]  <CNG>                       ;Z GRAD REFOCUSSING
         2M                                            ;AND PHASE ENCODING ON
         3M        [XT+X]  <CNG>                       ;ALL GRADS OFF
         1M
         3M        [XT+X]  <CNG TCHK>                  ;READOUT ON, TIMING CHECK
         AQ        [STA]                               ;START ACQUISITION
         2M        [XT+X]  <CNG>                       ;FREQ COMP FOR NEXT ECHO
         1M
         3M        [XT+X]  <CNG>                       ;ALL GRADS OFF
         2M
        END LOOP
        1M       [XT+X]  <CNG>                         ;STOP POINT
        D11
;
        GOTO START
;


------ FILE:  IMFLG.EXE

@DF
;
; X-GRADIENT FOR SW = 20.83 KHZ
;
; 1. ZOOM
DEFG GPHMERXC.021
4
F,0,F,-681,F,0,F,0
5
F,0,F,0,F,-317,F,-200,F,0
1,F,0
;
; 1.5 ZOOM
DEFG GPHMERXC.022
4
F,0,F,-970,F,0,F,0
5
F,0,F,0,F,-476,F,-158,F,0
1,F,0
;
; 2. ZOOM
DEFG GPHMERXC.023
4
F,0,F,-1228,F,0,F,0
5
F,0,F,0,F,-616,F,-270,F,0
1,F,0
;
; Y-GRADIENT FOR 128 STEPS
;
; 1. ZOOM
DEFG GPHMEPYC.021
5
```

```
F,0,F,0,F,0,F,0,R,256,-4,0
5
F,0,F,0,F,0,F,0,F,0
0
;
; 1.5 ZOOM
DEFG GPHMEPYC.022
5
F,0,F,0,F,0,F,0,R,384,-6,0
5
F,0,F,0,F,0,F,0,F,0
0
;
; 2. ZOOM
DEFG GPHMEPYC.023
5
F,0,F,0,F,0,F,0,R,512,-6,0
5
F,0,F,0,F,0,F,0,F,0
0
;
; Z GRADIENT
;
; GOOD SLICE PAIRS ARE:
; 500,600 GIVING 2.73MM SLICE
; 400,???
; 300,???
; 270,???      "     5MM SLICE
; 135,???      "     10MM SLICE
; 90,118       "     15MM SLICE
; 68,???       "     20MM SLICE
; 54,71        "     25MM SLICE
;
DEFG GPHMESZC.021
5
F,500,F,-400,F,400,F,0,F,600
5
F,0,F,0,F,0,F,0,F,0
0
;
DEFG GPHMESZC.022
5
F,220,F,-400,F,400,F,0,F,280
5
F,0,F,0,F,0,F,0,F,0
0
;
DEFG GPHMESZC.023
5
F,147,F,-400,F,400,F,0,F,190
5
F,0,F,0,F,0,F,0,F,0
0
;
DEFG GPHMESZC.024
5
F,90,F,-400,F,400,F,0,F,118
5
F,0,F,0,F,0,F,0,F,0
0
;
DEFG GPHMESZC.025
```

```
5
F,54,F,-400,F,400,F,0,F,71
5
F,0,F,0,F,0,F,0,F,0
0
@DN
```

Appendix H

The command file IMFLG (IMaging of FLow, Go) plus the pulse program execution file IMGEMGO.

```
------ FILE:  IMFLG

; DATE OF LAST EDITTING            5.10.86
;
@P
        ENTER THE NUMBER OF ECHOES ( 1, 2, 4, 8 )
@
@$T2
@DF
REG IMGEMTR
GY,,ZERO.001
BSI,512W
RECO,1,128,@$P2,1,1,1,-1,0,1
NE,4,D-,D11,.5S,NI,@$P2,NA,2,C2,@$P2
FL FQLIST,@CTL
@DN@RN
AU IMGEMGO,AP,,,AU
@P
SPECIFY REPETITION TIME ( IN S )
(TERMINATE WITH "S" FOLLOWED BY "RET" ) @@
D11@G,
@P
        ENTER THE NUMBER OF SCANS TO CHECK THE PROFILES
                                                        <RET>
@
@$T1
NE,@$P1
AI,0,AI,1,ZE
AU
@P
        CHECK ECHO POSITIONS
        ( 256, 512, 1280, 1792,    2304, 2816, 3328, 3840 )
                                                        <RET>
@
EP@G
@P

        PROFILE OF FIRST ECHO
@
SI,512W
STRT,0,BC,FT,MC,@W@DF
STRT 512W,BC,FT,MC
STRT,1KBC,FT,MC
STRT,1536WBC,FT,MC
@CTY
AI,0,NE,128
@P

TO ACQUIRE IMAGE START EXECUTION:
```

```
IMFL1    128 GRADIENT STEPS
@
STRT,0,SI,2

------ FILE:  IMGEMGO

TF
VDS 10
O1EN
STRT 0
LG
SI 1B
1         GOR
          DISP
IN 1
STPP
RA
O1DI
O1
SI 1N
EXIT
```

Appendix I

The command file IMFLTR (IMaging of FLow, gradient TRim) and its associated support files.

```
------ FILE: IMFLTR

2
@P
        WHICH GRADIENT TO USE FOR SLICE ?

        GX, GY, GZ, OR <RET> FOR NONE: @@$T6
EXE IMFLTRR

------ FILE: IMFLTRR

2
@P

        TO QUIT TYPE Q AT END FOLLOWED BY CNTL K.
        IF EXIT OTHER THAN THIS IMGETR.001 IS LOST
        FROM DISK. REWRITE IT FROM JOB TO DISK (WR).

@
@P

        WHICH POINT TO BE TRIMMED?

        READ DIRECTION (X,Y,Z): @@$T4
;
; NOW GO AND GET THE READOUT GRADIENT STRENGTH
;
@P
        ENTER THE READOUT GRADIENT STRENGTH
        ( I.E., THE ACQUISITION GRADIENT )
@
;
; CHECK IF FILE THERE
;
TSTF IMFLTR@$P4,ED,IMFLTR@$P4,,,,,,,,,,,@G@EQ,EXE IMFLTRG
@P
        SORRY, THAT GRADIENT CANNOT YET BE
        USED AS A READOUT GRADIENT...
        BETTER LUCK NEXT TIME
@

------ FILE: IMFLTRG

2
;
; GET THE GRADIENT PULSE TO TRIM
;
@P
```

```
CHOOSE A READOUT DIRECTION GRADIENT PULSE
                    TO ADJUST

        FIRST  READ DEPHASING:          R1
        2ND ECHO    READ COMP:          R2
        3RD ECHO    READ COMP:          R3
                        .
                        .
                        .
        8TH ECHO    READ COMP:          R8

                 INPUT:  @@$T5
NE,4
TSTF IMFLT@$P5@$P4,EXE IMFLT@$P5@$P4
@P

        KANADISCHE IDIOT
@


------ FILE:  IMFLTR1X

RECO,1,128,1,1,1,1,-1,0,1
C2,1,NI,1
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,@G@EQ
EXE IMFLTRX


------ FILE:  IMFLTR2X

RECO,1,128,2,1,1,1,-1,0,1
C2,2,NI,2
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX


------ FILE:  IMFLTR3X

RECO,1,128,3,1,1,1,-1,0,1
C2,3,NI,3
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX


------ FILE:  IMFLTR4X

RECO,1,128,4,1,1,1,-1,0,1
C2,4,NI,4
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX


------ FILE:  IMFLTR5X

RECO,1,128,5,1,1,1,-1,0,1
C2,5,NI,5
```

```
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX

------ FILE:  IMFLTR6X

RECO,1,128,6,1,1,1,-1,0,1
C2,6,NI,6
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX

------ FILE:  IMFLTR7X

RECO,1,128,7,1,1,1,-1,0,1
C2,7,NI,7
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX

------ FILE:  IMFLTR8X

RECO,1,128,8,1,1,1,-1,0,1
C2,8,NI,8
GX,DGY,DGZ,D
G@$P4,PGPHRTRIM.001
ED,IMFLTRX,,,,,,,,,,,,,,,@G@EQ
EXE IMFLTRX

------ FILE:  IMFLTRX

2@DF@RF
; X-GRADIENT FOR SW = 25 KHZ
;
; 2. ZOOM
DEL GPHRTRIM.001
DEFG GPHRTRIM.001
4,F,0,F
-970
F,0,F,0
5
F,0,F,0,F
-476
F
-159
F,0
1,F,0
2SI,4,ZE
TSTS,@$P6,GZ,GZ,P,GPHMESZC.024
TSTS,@$P6,GY,GY,P,GPHMESZC.024
TSTS,@$P6,GX,GX,P,GPHMESZC.024
@DN@RN2AU IMGEMGO
@DF@RF
AI,1
2SI,512W
TSTS,@$P5,R1,STRT,0
TSTS,@$P5,R2,STRT,512W
```

```
TSTS,@$P5,R3,STRT,1
TSTS,@$P5,R4,STRT,1536W
TSTS,@$P5,R5,STRT,2
TSTS,@$P5,R6,STRT,2560W
TSTS,@$P5,R7,STRT,3
TSTS,@$P5,R8,STRT,3584W
;
2EBC,SRV,FT,APK
;
@DN@RN
@P
        FIND THE MINIMUM ABSOLUTE VALUE OF "1PHZ"
@
@DBTY,@DB@RF@P

                                        TYPE
        TRY ANOTHER VALUE                T
        QUIT                             Q

                              INPUT:  @@$T9
TSTS,@$P9,T,EXE IMFLT@$P5@$P4
2EXE IMFLTRG
```

Appendix J

The command file IMFL1 (IMaging of FLow, 128 phase encoding steps) and the imaging execution file IMGEMI1.

```
------ FILE: IMFL1

@DF
NE,128
REG IMGEMTR
@P
        ENTER THE FILENAME ( NO EXTENSION )
        ( TERMINATE WITH A <RET> )
@
@$T3
@DN
AU IMGEMI1,@$P3
FL FQZERO.001,@CTL
AU RECO,@$P3
```

```
------ FILE: IMGEMI1

; AUTOMATION FOR 2DFT IMAGING: HIGH POSER VERSION
;
; MULTI ECHOS             128 GRADIENT STEPS
TF
O1EN
VDS 9
LG
STRT 0
SI 1N
WR #1 CR=128
SI 1A
10      GOD
        LO 10 TIMES 6
SI 1N
1       TOGL
        SI 1A
        GOR
        SI 1N
        WR #1 IB NW
IN 1
STPP
RA
SI 1B
EXIT
```

Appendix K

The phase correction program PHASCOR.

```
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
ccccccccccC
C
        PROGRAM PHASCOR
C
C       Steven Knudsen
C       Last Edit: 3/09/86
C       10-102 Clinical Sciences Bldg
C
C This program allows interactive correction of some phase errors
C through an examination of the time domain data from an image.
C
C A subroutine to analyze the phase of one or more echos in an image is used.
C The phase of an echo may be offset if that echo is not properly
C set in the middle of a sample window.
C
C If h(k) is the sampled signal for k = 0,1,2,...,N-1 where N is the total
C number of samples and H(n) is the Discrete Fourier Transform ( DFT )
C of h(k);
C
C       i.e.      h(k) <=> H(n)
C
C then the Time Shift Theorem says that;
C
C                 h(k-i) <=> H(n) * exp( j*2*pi*n*i/N )
C
C                         where i is the number of time shifted points in the
C                                 sample window
C
C We see that a shift of i=1 gives a phase shift of 2*pi across the whole
C spectrum - i=2 a shift of 4*pi, etc.
C
C Since the phase of the spectrum may be affected by other things ( such as
C movement of the spins during the scan ) I can think of no satisfactory
C computer algorithm to detect this shift. Therefore I have decided to let
C some user find this shift through inspection of the phase spectrum of a
C middle echo  for each set of echos. For each echo the number of shifted
C points is specified and written to a file for later use by the reconstruction
C program RECO.
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
cccccccccC
C
C Modified by S Knudsen              8-Jul-86
C
C Now uses array processor to unscramble the data whenever possible
C
ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccC
cccccccccC
C
        IMPLICIT INTEGER*4 (I-N)
        INTEGER*4 BUFF(128),N_BLK,M_BLK
        CHARACTER*1 SAMPLE,ALREADY
        LOGICAL*1 MATTYP,INNAM(60),ACCNAM(15),TMPNAM(60)
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,NECHO
        COMMON /BB/ BUFF,N_BLK,M_BLK,SAMPLE,ALREADY
        DATA ACCNAM/'[','S','T','E','V','E','N','.','A','S',
        1   'P','E','C','T',']'/
C--------------------------------------------------------
        TYPE 100
100     FORMAT('0** IMAGE PHASE CORRECTION PROGRAM ')
C--------------------------------------------------------
C Read file names
C--------------------------------------------------------
1       TYPE 102
102     FORMAT(1X,/,'$Enter input file name of the raw data file:  ')
        DO 3 I=1,60
3       TMPNAM(I)=0
        ACCEPT 104,TMPNAM
104     FORMAT(60A1)
        IF(TMPNAM(1).EQ.' ') STOP
C
        IFLG=0
        DO 4 I=1,60
        IF(TMPNAM(I).EQ.'['.OR.TMPNAM(I).EQ.':') IFLG=-1
4       CONTINUE
        IF(IFLG.NE.0) THEN
          DO 5 I=1,60
5         INNAM(I)=TMPNAM(I)
        ELSE
          DO 7 I=1,15
7         INNAM(I)=ACCNAM(I)
          DO 9 I=16,60
          IM=I-15
9         INNAM(I)=TMPNAM(IM)
        END IF
        TYPE 111,INNAM
111     FORMAT(' ',60A)
        DO 10 I=60,1,-1
        IF(INNAM(I).NE.' ') GO TO 12
10      INNAM(I)=0
C--------------------------------------------------------
 12     OPEN(UNIT=1,FILE=INNAM,ACCESS='DIRECT',
   *  STATUS='OLD',RECL=128,ASSOCIATEVARIABLE=N_BLK,ERR=900)
C--------------------------------------------------------
14      TYPE 106
106     FORMAT(1X,/,'$Enter number of frequency points (64, ',
        1   '128 or 256): ')
        ACCEPT *,NFREQ
        IF(NFREQ.NE.64.AND.NFREQ.NE.128.AND.NFREQ.NE.256)
        1   GO TO 14
C
16      TYPE 107
107     FORMAT(1X,/,'$Enter number of phase points (64, ',
        1   '128 or 256): ')
        ACCEPT *,NPHASE
        IF(NPHASE.NE.64.AND.NPHASE.NE.128.AND.NPHASE.NE.256)
        1   GO TO 16
C
        TYPE *,' '
17      TYPE *,'Enter the number of echos ( 1, 2, 4, 8, 16 ):'
        ACCEPT *,NECHO
        IF(NECHO.NE.1.AND.NECHO.NE.2.AND.NECHO.NE.4) THEN
          IF(NECHO.NE.8.AND.NECHO.NE.16) GOTO 17
        END IF
C
```

```
        IF(NPHASE.GT.NFREQ) GO TO 910
        NFLG=NFREQ
        IF(NFLG.EQ.64) THEN
          NBLOCK=16
          MATTYP='4'
        ELSE IF(NFLG.EQ.128) THEN
          NBLOCK=64
          MATTYP='6'
        ELSE
          NBLOCK=256
          MATTYP='8'
        END IF
C-----------------------------------------------------------
C Read input file, sort data from INTEGER*3 to INTEGER*4 (ie.
C from Aspect to VAX/PDP format), and write to temporary output
C file (unit = 4).  Close input file (unit = 1).
C-----------------------------------------------------------
        CALL UNSCRAM
C-----------------------------------------------------------
C Read temporary file (unit = 4) into virtual array.
C Read the middle echo for each set of echos.
C DFT each echo, inspect the phase of the spectrum and make the
C necessary corrections. Check corrections, repeat if necessary
C and save the corrections in a file.
C Finally, close all files.
C-----------------------------------------------------------
        CALL FOURIER
C-----------------------------------------------------------
C-----------------------------------------------------------
        TYPE 108
108     FORMAT('0*** Corrections Completed! ***',/)
        GO TO 1
C-----------------------------------------------------------
C Error routines
C-----------------------------------------------------------
900     TYPE 902
902     FORMAT('0** ERROR ** Unable to open input image file')
        CLOSE (UNIT=1)
        STOP
C-----------------------------------------------------------
910     TYPE 912
912     FORMAT('0** ERROR ** The number of frequency points '
        1     'must be greater than ',/,13x,
        2     'or equal to the number of phase points!',/)
        CLOSE (UNIT=1)
        STOP
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCC
C
        SUBROUTINE UNSCRAM
C
C       THIS PROGRAM UNSCRAMBLES "FILE_NAME" (TRANSFFERED FROM ASPECT
C     USING "MSPR")
C       THE OUTPUT FILE WILL HAVE THE SAME NAME AS THE INPUT FILE AND
C       .DAT EXTENSION.
C       THE PROGRAM MUST BE RUN USING  $SYS$SYSDEVICE:UNSCRAM file_name.ext
C
C
C       IN_BYTES        1 2 3...4 5 6..........
C                       | | |  | | |
C                       V V V  | V V
C       OUT_BYTES       1 2 3 4| 5 6 7 8 ...
```

```
C                                  ^|_________^ ^
C                                  |_____sign____|
C
C       VER #1  MAY,30 1984       U&A, APPLIED SCIENCES IN MEDICINE
C                                  DAN GHEORGHIU
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C S Knudsen              28-May-86
C
C Modified for use in 2dfft reconstruction routine 'phas2dap'.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C S Knudsen              8-Jul-86
C
C Modified to use the array processor
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        IMPLICIT INTEGER*4 (A-Z)
C
        BYTE            IN_BUFFER(512)
        BYTE            STACK(4),TEMP
        BYTE            LONG_BUFFER(1024)
        INTEGER*4       OUT_BUFFER4(128),IN_BUFFER4(128),BUFF(128)
        INTEGER*4       IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE, F_SIZE
C
        INTEGER*4 ACQ_FLAG, ACQ_MASK
        LOGICAL*1 MATTYP, SAMP_MODE
        CHARACTER*1 YES, SAMPLE,ALREADY
C
        COMMON /BB/ BUFF,N_BLK,M_BLK,SAMPLE,ALREADY
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,NECHO
        COMMON /MODE/ IMODE, ICODE, MSP_ERR
        COMMON /UNSC/          IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
C
        EQUIVALENCE     (STACK,STACK4)
        EQUIVALENCE     (IN_BUFFER,IN_BUFFER4)
        EQUIVALENCE     (LONG_BUFFER,F_SIZE)
C-----------------------------------------------
        BYTE            STRING(4)/27,'[','6','n'/
        BYTE            BELL/7/
        BYTE            REPORT(20)
        INTEGER*2       IOSB(4)
        INTEGER*4       SYS$ASSIGN,SYS$QIO,SYS$WAITFR
        INTEGER*4       TERMIN_MASK,FUNC_CODE
C
        INCLUDE                 '($IODEF)'
        INCLUDE                 '($SSDEF)'
C-------------------------------------------------
        DATA    IMODE/3/
        DATA ACQ_MASK/'000000C0'X/
        SAMP_MODE = .TRUE.
        IN_CHANNEL = 1
        OUT_CHANNEL = 4
C
C-------------------------------------------------------
C
 92     TYPE *,' ALREADY UNSCRAMBLED ( Y OR N )? '
        READ(5,1000) ALREADY
        IF(ALREADY.NE.'Y'.AND.ALREADY.NE.'N') GOTO 92
C
C
        OPEN(UNIT=4,FILE='[STEVEN.ASPECT]DUMDUM.TP1',
        1          ACCESS='DIRECT',STATUS='OLD',RECL=128,
        2          ASSOCIATEVARIABLE=M_BLK,ERR=930)
 101    CONTINUE
        IF(ALREADY.EQ.'Y') THEN
          TYPE *,' Sample mode ( M or Q )'
          READ(5,1000) SAMPLE
          IF(SAMPLE.NE.'M'.AND.SAMPLE.NE.'Q') GOTO 101
          CLOSE(UNIT=1)
          RETURN
        END IF
C-----------------------------------------------------------
C
C check out the first block for aquisition parameters, namely SEQ or SIM
C sampling
C
C
C       GET THE FILE_SIZE IN BYTES AND TRANSFER HEADER BLOCK AS IT IS.
C       - 1 LONG_WORD = FILE_SIZS/3
C       - 648 BYTES (216 ASPECT_WORDS) USER INFO
C
        N_BLK=1                                 ! BLOCK VARIABLE OF INPUT BUFFER
        M_BLK=1                                 ! BLOCK VARIABLE OF OUTPUT BUFFER
C
        READ(1'N_BLK) (LONG_BUFFER(I),I=1,512)
        READ(1'N_BLK) (LONG_BUFFER(I),I=513,1024)
        FILE_SIZE=F_SIZE*3
C
C
        OUT_128=1                               ! OUTPUT BUFFER COUNTER
        OUT_BUFFER4(1)=FILE_SIZE
        L=0                                     ! BYTE COUNTER OF DATUS
        J=0                                     ! POINTS EVEN DATUS
C
D       TYPE *,'FILE_SIZE=',FILE_SIZE
C
        DO 110 N_BYTES=5,664
C
                L=1+JMOD(L,3)                   ! INCREMENT BYTE PTR OF DATUS
                STACK(L)=LONG_BUFFER(N_BYTES)           ! COPY UP TO 3 BYTES
                IF(L.EQ.3) THEN
                        J=1+JMOD(J,2)           ! INCREMENT EVEN PTR OF DATA
                        IF(J.EQ.2) THEN         ! EVEN DATUS NEEDS SWAPING
                                TEMP=STACK(1) ! PERMUTE 3 BYTE DATUS
                                STACK(1)=STACK(2)
                                STACK(2)=STACK(3)
                                STACK(3)=TEMP
                        ENDIF
C
                        STACK(4)=0
C
C                       CHECK IF OUTPUT_BUFFER IS FULL
C
                        OUT_128=1+JMOD(OUT_128,128) ! OUTPUT COUNTER INCREMENT
                        OUT_BUFFER4(OUT_128)=STACK4
                        IF(OUT_128.EQ.128) THEN
D                               TYPE *,' M_BLK=  ',M_BLK
                                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
                                IF(SAMP_MODE) THEN
                                  ACQ_FLAG = ACQ_MASK.AND.OUT_BUFFER4(45)
                                  SAMP_MODE = .FALSE.
                                END IF
                        ENDIF
C
                ENDIF
C
110     CONTINUE
C
        IF(OUT_128 .NE. 128) THEN
                OUT_128  = OUT_128 +1
                DO J=OUT_128 ,128
                        OUT_BUFFER4(J)=0
                ENDDO
                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
        ENDIF
C
C find out what type of sampling was used
C
        IF(ACQ_FLAG.EQ.'000000C0'X) THEN
          TYPE *,'The data is sampled simultaneously'
          SAMPLE = 'M'
        ELSE
          TYPE *,'The data is sampled sequentially'
          SAMPLE = 'Q'
        END IF
C
C       Call MSP, if not busy !
C
        CALL MSATT_UNS(0,ISTATUS)       !copy after Fipak's MSATT
C                                               (returns MSP status)
D       TYPE *,'ISTATUS=',ISTATUS
        IF(ISTATUS .EQ. SS$_NORMAL) THEN
                ASP_FILE_SIZE = FILE_SIZE/3
                CALL UNSCRAM_WITH_MSP
                IF(MSP_ERR .NE. 1) GOTO 80
        ENDIF
C
C       MSP busy, let's do it on VAX
C
C
C       CONVERT THE REST OF DATA
C       #ASP_WORDS=#VAX_LONG_WORDS
C
        IN_512=0                                !INPUT BUFFER COUNTER
        OUT_128=0                               !OUTPUT BUFFER COUNTER
        L=0                                     !BYTE COUNTER OF DATUS
        J=0                                     !POINTS EVEN DATUS
C
        DO 10 N_BYTES=1,FILE_SIZE
C
                IN_512=1+JMOD(IN_512,512)
                IF(IN_512.EQ.1) READ(1'N_BLK) (IN_BUFFER4(I),I=1,128)
D       TYPE *,'N_BLK=',N_BLK
                L=1+JMOD(L,3)                   !INCREMENT BYTE PTR OF DATUS
                STACK(L)=IN_BUFFER(IN_512)      !COPY UP TO 3 BYTES
                IF(L.EQ.3) THEN
                        J=1+JMOD(J,2)           !INCREMENT EVEN PTR OF DATA
                        IF(J.EQ.2) THEN         !EVEN DATUS NEEDS SWAPING
                                TEMP=STACK(1)   !PERMUTE 3 BYTE DATUS
                                STACK(1)=STACK(2)
                                STACK(2)=STACK(3)
                                STACK(3)=TEMP
                        ENDIF
C
                        IF(STACK(3).LT.0) THEN  !TAKE CARE OF SIGN
```

```
                                STACK(4)=-1
                        ELSE
                                STACK(4)=0
                        ENDIF
C
C                       CHECK IF OUTPUT_BUFFER IS FULL
C
                        OUT_128=1+JMOD(OUT_128,128)  ! OUTPUT COUNTER INCREMENT
                        OUT_BUFFER4(OUT_128)=STACK4
                        IF(OUT_128.EQ.128) THEN
D                               TYPE *,' M_BLK=  ',M_BLK
                                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
                        ENDIF
C
                ENDIF
C
10      CONTINUE
C
C       WRITE THE LAST BLOCK IF ANY
C
130     IF(OUT_128.NE.128) THEN
D               TYPE *,' LAST M_BLK=  ',M_BLK
                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
        ENDIF
C
C       NOW CLOSE ALL FILES AND EXIT.
C
80      CLOSE(UNIT=1)
C
        TYPE *,' OUT OF UNSCRAM'
        RETURN
 1000   FORMAT(A)
C
C-------------------------------------------------------------
C Error routines
C-------------------------------------------------------------
930     CONTINUE
        OPEN(UNIT=4,FILE='[STEVEN.ASPECT]DUMDUM.TP1',
        1       ACCESS='DIRECT',STATUS='NEW',RECL=128,
        2       ASSOCIATEVARIABLE=M_BLK,ERR=931)
        GOTO 101
931     TYPE 932
932     FORMAT('0** ERROR ** Unable to open file DUMDUM.TP1')
        GOTO 999
C
950     TYPE 952
952     FORMAT('0** ERROR ** Error reading input image file')
        GOTO 999
985     TYPE 987
987     FORMAT('0** ERROR ** Error writing file DUMDUM.TP1')
        GOTO 999
C
999     CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        CLOSE (UNIT=4)
        STOP
        END
C
C********************************************************************
C
        SUBROUTINE     UNSCRAM_WITH_MSP
C
C       Subroutine assumes that the header block has been copied !
```

```
C    For more detail see MSP array processor manual and/or talk
C         with system manager
C
C         Dan G.
C-----------------------------------------------------------------------
C
          INTEGER*4        IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
          INTEGER*4        MSP_REG(16)
          INTEGER*4        MSP_DRAM        /'400000'X/
          INTEGER*4        MSP_PROGRAM /'C01400'X/
          INTEGER*4        MSP_START       /'C0140C'X/
          INTEGER*4        BUFFER(128)
          INTEGER*4        ICHANNEL
C
          COMMON /MODE/          IMODE,ICODE,MSP_ERR
          COMMON /UNSC/          IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
C
D         TYPE *,'IMODE ',IMODE
D         IMODE_OLD=IMODE
C
          CALL MSCLR                         !CLEAR MSP
          IF(IWAIT('MSCLR ').NE.0)GO TO 92
          CALL ENBDMA                        !ENABLE DMA
C
C         Load mini_program
C
          ICHANNEL=3
          CALL ASSIGN(ICHANNEL,'DISK$USER:[STEVEN.ASPECT]UNSCMISAL.EXE')
          NBLKS=3
          NBYTES=512
          CALL VMNLOAD(ICHANNEL,MSP_PROGRAM,NBLKS,NBYTES)            !LOAD MINI
PROGRAM
          IF(IWAIT('MNLOAD').NE.0)GO TO 92
          CALL CLOSE(ICHANNEL)
C
          CALL PSWBSH('1'X)                           !Disable FPP interupt
          IF(IWAIT('PSWBSH').NE.0)GO TO 92
C
          N_BLK = 3
          M_BLK = 3
          N_DATA= 0                                   !general data_counter
          N_MAX_MSP='20000'X                          !MSP_memory capacity
          MSP_REG(11) = MSP_DRAM                                !R2 <= unsc_data_address
          MSP_REG(16) = MSP_DRAM + '20000'X           !R2 <= sc_data_address
C
          DO WHILE( N_DATA .LT. ASP_FILE_SIZE )
                N_DATA_MSP = ASP_FILE_SIZE - N_DATA
                IF( N_DATA_MSP .GT. N_MAX_MSP) N_DATA_MSP = N_MAX_MSP
C
C               Read input file and put data into MSP
C
                N_LOOP= 0
                N_MSP = 0
                MSP_ADDR= MSP_DRAM + '20000'X          !start loading here
D               TYPE 85,MSP_ADDR
D               IMODE=3
85              FORMAT(' ',4Z12)
c
                DO WHILE(N_MSP .LT. N_DATA_MSP)
                      READ (IN_CHANNEL'N_BLK,ERR=100) BUFFER
                      CALL      VPUT(BUFFER,MSP_ADDR,4,128)
                      MSP_ADDR= MSP_ADDR +512
                      N_LOOP = N_LOOP + 1
                      N_MSP = (512*N_LOOP)/3
```

```
CD                      TYPE 85,MSP_ADDR,N_BLK,N_LOOP,N_MSP
                        N_BLK = N_BLK + 1
                        IF          (IWAIT('BUFFER').NE.0)    GO TO 92
                ENDDO
D               TYPE 85,MSP_ADDR
D               IMODE=IMODE_OLD
C
C               Load MSP registers with "unscrambling" parameters
c
C
                CALL MSCLR                              !CLEAR MSP
                IF(IWAIT('MSCLR ').NE.0)GO TO 92
                CALL PSWBSH('1'X)                               !Disable FPP interupt
                IF(IWAIT('PSWBSH').NE.0)GO TO 92
C
                MSP_REG(13) = N_DATA_MSP                        !R4 <= counter
D               TYPE 85,MSP_REG(11),MSP_REG(13),MSP_REG(16)
C
                CALL    MSWRTE('9400'X,MSP_REG)                                 !R2,R4,R7
                IF      (IWAIT('WR9400').NE.0)GO TO 92
D2000   CONTINUE        !after break
C
                CALL    VMNSTRT ( MSP_START )
                IF      (IWAIT('STUNSC').NE.0)GO TO 92
C
C               Retrieve unscrambled_data from MSP into output_file
C
                M_MSP = 0
                MSP_ADDR= MSP_DRAM
D               TYPE 85,MSP_ADDR
D               IMODE=3
                DO WHILE(M_MSP .LT. N_DATA_MSP)
                        CALL    VGET(BUFFER,MSP_ADDR,4,128)
                        M_MSP = M_MSP + 128
                        IF(IWAIT('BUFFER').NE.0) GO TO 92
C
                        IF(M_MSP .GT. N_DATA_MSP) THEN ! zero_filling in last
                                M_FILL_ZERO=1+MOD(N_DATA_MSP,128)
                                DO JJ=M_FILL_ZERO,128  ! block ,following data.
                                        BUFFER(JJ)=0
                                ENDDO
                        ENDIF
C
                        WRITE (OUT_CHANNEL'M_BLK,ERR=110) BUFFER
                        M_BLK = M_BLK +1
                        MSP_ADDR= MSP_ADDR +512
                ENDDO
D               TYPE 85,MSP_ADDR
D               IMODE=IMODE_OLD
C
                N_DATA = N_DATA + N_DATA_MSP
        ENDDO ! N_DATA loop
C
92      CALL MSDET
        RETURN
C
100     TYPE *,'*** ERROR  reading input_file '
        GOTO 92
C
110     TYPE *,'*** ERROR  writing output_file '
        GOTO 92
C
        END
CC*********************************************************************
```

```
      SUBROUTINE MSATT_UNS (LUN,ISTATUS)
C
C
C     Copyright (c) 1982,1983
C
C     Computer Design and Applications, Inc.,
C     411 Waverly Oaks Road
C     Waltham, Massachusetts 02154
C
C     All rights reserved.
C
C
C     MSATT - Attach MSP-3000 to task
C
C     This subroutine attaches the MSP-3000 to the task for
C     exclusive use. No other user may attach it while it is
C     in use.
C
C
C     Version 0.0             15-Feb-83         Peter Kapinos
C
C                             First Release
C
C
      IMPLICIT                NONE
C
      INCLUDE                        '($SSDEF)'
      INCLUDE                        '($IODEF)'
      INTEGER*4               LUN,ISTATUS
      INTEGER*4               SERVICE_STATUS
      INTEGER*4               SYS$ASSIGN
      INTEGER*4               SYS$SETEF
      INTEGER*4               FINAL_IO_STATUS
      INTEGER*4               SYS$QIOW
      INTEGER*4               IO$_ALTDMA
      INTEGER*4               BITS_TO_SET
      INTEGER*4               BITS_TO_CLEAR
      CHARACTER*5             M3K_NAME
      LOGICAL*1               FPP_SAFE
      INTEGER*2               IOSB (4)
      INTEGER*4               EVENT_FLAG,M3K_CHANNEL
C
      COMMON         /QIO_ARGUMENTS/         EVENT_FLAG,M3K_CHANNEL,IOSB
      COMMON         /SAFE_MODE/    FPP_SAFE
C
C     Establish the MSP-3000 device name for the assign
C
      PARAMETER               (M3K_NAME   = 'CDA0:')
C
C     Equate MSP-3000 $QIO function code with existing DEC function code
C
      PARAMETER               (IO$_ALTDMA = IO$_ACCESS)
      PARAMETER               (BITS_TO_SET = 0)
      PARAMETER               (BITS_TO_CLEAR = 'FFFFFFFF'X)
C
C     Start of executable code
C     ----------------------
C
C     Establish the event flag number for all system services
C
      EVENT_FLAG = 32
C
C     Default mode at attach is safe
C
```

```
        FPP_SAFE = .TRUE.
C
C       Set the event flag so the first MSWAIT works correctly
C
        SERVICE_STATUS          = SYS$SETEF (%VAL (EVENT_FLAG))
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .EQ. SS$_WASSET) GO TO 400
C
        IF (SERVICE_STATUS .EQ. SS$_WASCLR) GO TO 400
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,310) SERVICE_STATUS
310       FORMAT (' MSATT SETEF Failed with code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (SERVICE_STATUS)
C
                  ISTATUS=SERVICE_STATUS
                  RETURN
        END IF
C
C       Assign the MSP-3000 to this task
C
400     SERVICE_STATUS          = SYS$ASSIGN ( M3K_NAME,M3K_CHANNEL,,)
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,410) SERVICE_STATUS
410       FORMAT (' MSATT Assign Failed with code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (SERVICE_STATUS)
C
                  ISTATUS=SERVICE_STATUS
                  RETURN
        END IF
C
C       Issue QIO to clear the ALTDMA mask stored in the UCB
C
        SERVICE_STATUS          = SYS$QIOW ( %VAL (EVENT_FLAG), %VAL (M3K_CHANNEL),
     *                         %VAL (IO$_ALTDMA),IOSB,,,
     *               %VAL (BITS_TO_SET),
     *                                %VAL (BITS_TO_CLEAR),,,,)
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
```

```
C
        CALL BLANK (1)
C
        WRITE (5,510) SERVICE_STATUS
510     FORMAT (' MSATT $QIO Failed with code = ',I9)
C
C       This error is serious so exit from image
C
C       CALL EXIT (SERVICE_STATUS)
C
              ISTATUS=SERVICE_STATUS
              RETURN
       END IF
C
C      Check the I/O Status Block to see if there were any I/O errors
C
       FINAL_IO_STATUS = IOSB (1)
C
       IF (FINAL_IO_STATUS .NE. SS$_NORMAL) THEN
C
        CALL BLANK (1)
C
        WRITE (5,600) FINAL_IO_STATUS
600     FORMAT (' MSATT I/O Failed With Code = ',I9)
C
C       This error is serious so exit from image
C
C       CALL EXIT (FINAL_IO_STATUS)
C
              ISTATUS=SERVICE_STATUS
              RETURN
       END IF
C
C      Put a success code in I/O Status Block so first MSWAIT works
C      correctly
C
       IOSB (1) = SS$_NORMAL
       ISTATUS  = SS$_NORMAL
C
C      We're done here so return.
C
       RETURN
C
       END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
       SUBROUTINE FOURIER
C
C This routine reads in a specified echo set ( from the scan with
C no phase encoding pulse on ) and examines the phase of the
C particular echo of interest. Corrections are input by the user
C and the results checked graphically. When a satisfactory phase
C profile has been found,  the correction factor for that echo is
C saved. These factors are used later in the 2DFT reconstruction.
C
       IMPLICIT INTEGER*4 (I-N)
       LOGICAL*1 MATTYP,ISTRING(60)
       INTEGER*2 BUFFW(256),IWDUM,NSHIFT(32),NMULT(32)
       INTEGER*4 BUFFI(128), NECHO
       REAL*4 BUFFR(128)
       REAL*4 XR(512),XI(512),XPHAS(512),XMAG(512)
       COMPLEX*8 BUFFC(64),ZMAT(256,256)
```

```
      COMPLEX ZTEMPC(256),ZTEMC
      REAL ZTEMPR(512)
      CHARACTER*32 ANSWER
      CHARACTER*1 SAMPLE,ALREADY,NEXT,FINE_TUNE
C
C
      EQUIVALENCE (ZTEMPC(1),ZTEMPR(1))
      EQUIVALENCE (BUFFI(1),BUFFW(1),BUFFR(1),BUFFC(1))
C
C
      COMMON /BB/ BUFFW,N_BLK,M_BLK,SAMPLE,ALREADY
      COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,NECHO
      COMMON /FFTARR/ XR,XI
C
C process NECHO echos from the DUMDUM file
C
      DO NE = 1, NECHO
      TYPE *,'ECHO = ',NE
C--------------------------------------------------
C Zero virtual array
C--------------------------------------------------
       DO I = 1, 512
       ZTEMPR(I) = 0.0
       END DO
C
       DO 12 J=1,NFLG
       DO 10 I=1,NFLG
10     ZMAT(I,J)=CMPLX(0.0,0.0)
12     CONTINUE
C--------------------------------------------------
C Read file into virtual array. Remember that the echos are centred
C in the array.
C Note that if there is zero filling, the zeros are placed at
C the beginning and end of the matrix.
C--------------------------------------------------
C
       NFR=NFREQ/64
       NPHHA=NPHASE/2
       NFRHA=NFREQ/2
       NFLHA=NFLG/2
       M_BLK= ( NE - 1 )*NFR + 3                  !Remember 3 info blocks
       M_BSTEP = NFR * NECHO
C
       DO JJ=1,NPHASE
        M_BLKOLD = M_BLK
        IF(NPHASE.EQ.NFLG) THEN
         JPH=JJ
        ELSE
         JPH=JJ+NFLHA-NPHHA
        END IF
C
        DO KK=1,NFR
         READ (4'M_BLK,ERR=980) BUFFI
C
         DO I=1,128
          BUFFR(I)=REAL(BUFFI(I))      ! Convert from INT*4 to REAL*4
         END DO
C
         DO I=1,64                     !Write data to row of ZMAT
          IFR=((KK-1)*64)+I
          ZMAT(JPH,IFR)=BUFFC(I)
            END DO
        END DO
        M_BLK = M_BLKOLD + M_BSTEP
```

```
          END DO
C-------------------------------------------------------------
C
C At this point we have the echo data in the array ZMAT
C
C Now let's look at the middle + 1 scan in that image.
C
C copy it into the real and imaginary arrays used in the FFT
C
C for every loop this data array may need to be shifted n points
C right or left. This shift need be a logical shift
C
C         1 2 3 4 ...                   NFLG-1 NFLG
C i.e     +--+--+--+--+-         -+--+--+
C         | | | | | ...      | | |        right logical shift
C         +--+--+--+--+-         -+--+--+
C      ^ -> -> ->                             |
C         /\                                  |
C         |                                   |
C         +------------------------------+
C
C The variable ISHIFT indicates the direction and number of points to
C shift the time data
C                     ISHIFT = a negative integer then shift left
C                            = a positive  "    "    "  right
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        ISHIFT = 0
        NEXT = 'Y'
        IROW = NFLG/2+1
        DO WHILE(NEXT.EQ.'Y')
          DO I = 1,NFLG
            XR(I) = REAL(ZMAT(IROW,I))
            XI(I) = AIMAG(ZMAT(IROW,I))
          END DO
C
C get the number of points to shift left ( -ve integer ) or right
C ( +ve integer )
C
 1234   CONTINUE
        TYPE *,'Enter the number of points to shift the time domain data'
        TYPE *,' '
        TYPE *,'  a negative integer shifts the data left by that number'
        TYPE *,'  a positive integer shifts the data right by that number'
        TYPE *,' '
        ACCEPT *,ISHIFT
        IF(ABS(ISHIFT).GT.30) THEN
          TYPE *,'Oops. Too many points to shift the time data'
          TYPE *,'Better make a new image and centre your echos'
          TYPE *,' or reconsider your choice'
          GOTO 1234
        END IF
C
C shift the points as indicated by ISHIFT
C
          IF(ISHIFT.NE.0) THEN
            IF(ISHIFT.LT.0) THEN              !shift left
              DO I = 1,-1*ISHIFT
                  XRTEMP = XR(1)
                  XITEMP = XI(1)
                  DO J = 1,NFLG-1
                    XR(J) = XR(J+1)
                    XI(J) = XI(J+1)
            END DO
            XR(NFLG) = XRTEMP
            XI(NFLG) = XITEMP
         END DO
       ELSE                                  !shift right
        DO I = 1,ISHIFT
            XRTEMP = XR(NFLG)
            XITEMP = XI(NFLG)
            DO J = NFLG,2,-1
             XR(J) = XR(J-1)
             XI(J) = XI(J-1)
            END DO
            XR(1) = XRTEMP
            XI(1) = XITEMP
        END DO
       END IF
      END IF
C
C check the sampling technique
C
      IF(SAMPLE.EQ.'Q') THEN
C
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C see J.A. Lunt's famous 'Sampling Data the Bruker Way' for an explanation
C of what goes on when the data is sequentially sampled.
C+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
C
C negate every other complex point in the scan
C
       DO I = 2,NFLG,2
           XR(I) = XR(I)*-1.0
           XI(I) = XI(I)*-1.0
       END DO
C
C now treat all the points as real. Each imaginary point is considered
C to fall between the corresponding real point and the next real point
C
      J = 1
       DO I = 1,NFLG
        XPHAS(J) = XR(I)
        XPHAS(J+1) = XI(I)
           J = J + 2
       END DO
       DO I = 1,NFLG*2
           XR(I) = XPHAS(I)
           XI(I) = 0.0
       END DO
C
C and do a real fold about the centre of the frequency data
C This means a reversal of real and the old imaginary points about the
C centre of the time domain data
C
       DO I = 1,NFLG*2/4
           ZTT = XR(I)
           XR(I) = XR(NFLG*2/2+1-I)
           XR(NFLG*2/2+1-I) = ZTT
       END DO
       J = 1
       DO I = NFLG*2/2+1,NFLG*2/4*3
           ZTT = XR(I)
           XR(I) = XR(NFLG*2+1-J)
           XR(NFLG*2+1-J) = ZTT
        J = J + 1
       END DO
```

```
           END IF
C-----------------------------------------------------------
C Perform 2dft
C-----------------------------------------------------------
           IF(SAMPLE.EQ.'Q') THEN
             NFLG = NFLG*2
             CALL FFT1(NFLG)
           XNORM=1.0/NFLG                  ! Normalization factor
           ELSE
             CALL FFT1(NFLG)               ! Setup fft routine
           XNORM=1.0/NFLG                  ! Normalization factor
           END IF
C
           CALL FFT2
           IF(SAMPLE.EQ.'Q') NFLG = NFLG/2
           IF(SAMPLE.EQ.'M') THEN
             DO I=1,NFLHA          ! Reorder frequency terms
                   IP=I+NFLHA
                   DUM=XR(I)
                   XR(I)=XR(IP)
                   XR(IP)=DUM
                   DUM=XI(I)
                   XI(I)=XI(IP)
                   XI(IP)=DUM
             END DO
C
             DO I=1,NFLHA! and flip about x axis
                   IP=NFLG+1-I
                   DUM=XR(I)
                   XR(I)=XR(IP)
                   XR(IP)=DUM
                   DUM=XI(I)
                   XI(I)=XI(IP)
                   XI(IP)=DUM
             END DO
           END IF

C
C find the max and min magnitude in the spectrum
C
           XMAX = -1.E33
           XMIN = 1.E33
           DO I = 1,NFLG
             XT = SQRT(XR(I)*XR(I) + XI(I)*XI(I))
             IF(XT.LT.XMIN) XMIN = XT
             IF(XT.GT.XMAX) XMAX = XT
           END DO
C
C-----------------------------------------------------------
C
C now we can calculate the phase spectrum
C
C After the first time through the user is allowed to twiddle the
C phase spectrum.
C A phase correction term A + B*i is subtracted from each point i in
C the spectrum
C                  A = zero order term ( constant )
C
C                  B = first order term ( degrees / point )
C
C **** Note; in the current implementation the zero order term, A, is
C        fixed in magnitude ( i.e., A = 0 )
C-----------------------------------------------------------
C
```

```
       ZPHASE_AVG = 0.0
       ZPI = 3.1415926
C
C
       ZSLOPE = 0.0
       ZDELTA_SLOPE = ZSLOPE/FLOAT(NFLG)
       ZNOISE = 0.05*(XMAX - XMIN)
       XMAXP = -1.E33
       XMINP = 1.E33
       ZPHASESUM = 0.0
       IF(NE/2*2.EQ.NE) THEN
         ZFACT = -1.0
       ELSE
         ZFACT = 1.0
       END IF
       TYPE *,'the current echo is ',NE
       TYPE *,'the current ZFACT is ',ZFACT
       TYPE *,'Enter a multiplicative factor ( 1.0 OR -1.0 )'
       ACCEPT *,ZFACT
       DO I=1,NFLG
         XT = SQRT(XR(I)*XR(I) + XI(I)*XI(I))
         IF(ABS(XT).LT.ZNOISE) THEN
               XPHAS(I) = 0.0
         ELSE
               XR(I) = XR(I)*ZFACT
               XI(I) = XI(I)*ZFACT
               IF(XR(I).LE.1.E-20.AND.XR(I).GE.-1.E-20) XR(I)=1.0E-20
           TEMP = ATAN( XI(I)/XR(I) )                 ! find the phase angle
               IF(XR(I).LT.0.0.AND.XI(I).LT.0.0) TEMP=TEMP-ZPI
               IF(XR(I).LT.0.0.AND.XI(I).GT.0.0) TEMP=TEMP+ZPI
           IF(TEMP.GT.XMAXP) XMAXP=TEMP
           IF(TEMP.LT.XMINP) XMINP=TEMP
           XPHAS(I) = TEMP - ZDELTA_SLOPE*FLOAT(I) - ZPHASE_AVG
         END IF
         ZPHASESUM = ZPHASESUM + XPHAS(I)
       END DO
       ZPHASE_AVG = ZPHASESUM/FLOAT(NFLG)
       DO I = 1, NFLG
         XMAG(I) = SQRT( XR(I)*XR(I) + XI(I)*XI(I) )
       END DO
C
C plot this junk so the phase can be seen
C
       TYPE *,'MIN MAX PHASE = ',XMINP,XMAXP
       CALL PLOT( NFLG, XPHAS, XR, XI, XMAG )
C
 234   CONTINUE
       TYPE *,'Enter Y to have another look at this echo'
       TYPE *,'or N to proceed to the next echo'
       READ(5,1000) NEXT
       IF(NEXT.NE.'Y'.AND.NEXT.NE.'N') GOTO 234
      END DO
C
C save the shift value for this echo
C
      TYPE *,''
 345  CONTINUE
      WRITE(6,999) ISHIFT
      READ(5,1000) NEXT
      IF(NEXT.NE.'Y'.AND.NEXT.NE.'N') GOTO 345
      IF(NEXT.EQ.'Y') THEN
        NSHIFT(NE) = ISHIFT
        NMULT(NE) = INT(ZFACT)
      ELSE
```

```
          TYPE *,'Enter the correct shift for this echo'
          ACCEPT *,ISHIFT
          NSHIFT(NE) = ISHIFT
          NMULT(NE) = INT(ZFACT)
        END IF
C----------------------------------------------------------------------
        END DO                                    ! end of echo loop
C
C Echo the shift factors to the screen as a check
C
        WRITE(6,998)
        DO I = 1,NECHO
          WRITE(6,997) I,NSHIFT(I),NMULT(I)
        END DO
C
C Open unit 7 and write out the shift array and multiplicative array
C so that RECO can use it
C
        OPEN(UNIT=7,FILE='[STEVEN.ASPECT]SHIFTS.REC',FORM='UNFORMATTED',
        1        STATUS='UNKNOWN',ERR=931)
        WRITE(7) NSHIFT,NMULT
C
        CLOSE (UNIT=4)
        CLOSE ( UNIT=7 )
C
        RETURN
C
931     CONTINUE
        TYPE *,'Unable to open the SHIFTS file. Ack ! Pfffft...'
        STOP
997     FORMAT('  ',I2,9X,I4,10X,I4)
998     FORMAT(/,' ECHO   SHIFT FACTOR   MULT FACTOR',
        1   /'++++   ++++++++++++   +++++++++++')
999     FORMAT(/,' The current shift factor is',I3,'. Keep it, Y or N')
1000    FORMAT(A)
980     CONTINUE
        TYPE *,' Error opening raw data file. Bye bye.....'
        STOP
        END
C
C------------------------------------------------------------
C------------------------------------------------------------
   SUBROUTINE FFT1(NN)
C------------------------------------------------------------
C SUBROUTINE TO CALCULATE A FAST FOURIER TRANSFORM (FFT).
C THIS PROGRAM IS TAKEN FROM BRIGHAMM, E.O.: THE FAST FOURIER
C TRANSFORM, 164, PRENTICE-HALL INC., 1974.
C------------------------------------------------------------
   IMPLICIT INTEGER*4 (I-N)
   REAL*4 XR(512),XI(512),CC(512),SS(512)
   INTEGER*4 JP(512,16),JBITR(512)
   COMMON /FFTARR/ XR,XI
   DATA PI/3.14159265/
C------------------------------------------------------------
   IF(NN.LE.0.OR.NN.GT.512) GO TO 200
   DUM1=FLOAT(NN)
   DUM1=ALOG(DUM1)
   DUM2=ALOG(2.0)
   DUM2=DUM1/DUM2
   NU=NINT(DUM2)
   N=2**NU
   IF(N.NE.NN) GO TO 200
   CON=2.0*PI/N
C------------------------------------------------------------
```

```
C SETUP SINE AND COSINE VALUES.
C-------------------------------------------------------------------
   DO 12 NU1P1=1,NU
   NU1=NU1P1-1
   DO 10 KP1=1,N
   K=KP1-1
   JP(KP1,NU1P1)=(K/2**NU1)+1
 10 CONTINUE
 12 CONTINUE
C
   DO 14 JP1=1,N
   J=JP1-1
   JBITR(JP1)=IBITR(J,NU)
   ARG=CON*FLOAT(JBITR(JP1))
   CC(JP1)=COS(ARG)
   SS(JP1)=SIN(ARG)
 14 CONTINUE
   RETURN
C-------------------------------------------------------------------
C-------------------------------------------------------------------
   ENTRY FFT2
   N2=N/2
   NU1=NU-1
   K=0
   DO 100 L=1,NU
 102 DO 101 I=1,N2
   KP1=K+1
   NU1P1=NU1+1
   JP1=JP(KP1,NU1P1)
   C=CC(JP1)
   S=SS(JP1)
   K1=K+1
   K1N2=K1+N2
   TREAL=XR(K1N2)*C+XI(K1N2)*S
   TIMAG=XI(K1N2)*C-XR(K1N2)*S
   XR(K1N2)=XR(K1)-TREAL
   XI(K1N2)=XI(K1)-TIMAG
   XR(K1)=XR(K1)+TREAL
   XI(K1)=XI(K1)+TIMAG
 101 K=K+1
   K=K+N2
   IF(K.LT.N) GO TO 102
   K=0
   NU1=NU1-1
 100 N2=N2/2
   DO 103 K=1,N
   I=JBITR(K)+1
   IF(I.LE.K) GO TO 103
   TREAL=XR(K)
   XR(K)=XR(I)
   XR(I)=TREAL
   TIMAG=XI(K)
   XI(K)=XI(I)
   XI(I)=TIMAG
 103 CONTINUE
   RETURN
C-------------------------------------------------------------------
C ERROR ROUTINE
C-------------------------------------------------------------------
 200 TYPE 202
 202 FORMAT('0*** FFT SUBROUTINE ERROR - ILLEGAL "N" VALUE ***')
   STOP
   END
C-------------------------------------------------------------------
```

```
C-------------------------------------------------------------------
      FUNCTION IBITR(J,NU)
      IMPLICIT INTEGER*4 (I-N)
      J1=J
      IBITR=0
      DO 200 I=1,NU
      J2=J1/2
       IBITR=IBITR*2+(J1-2*J2)
  200 J1=J2
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
          SUBROUTINE PLOT( SIZE, PHAS, R, I, M )
C
C quick and dirty plotting of the data passed
C
C The output device used is a Digital VT241 with a colour monitor.
C See REGISUTL.FOR for the subroutine calls
C
          INTEGER*4 SIZE
          REAL*4 PHAS(512), R(512), I(512), M(512)
          INTEGER*2 X, Y, OLD_Y, OLD_X
          INTEGER*2 TT_CHAN
          INTEGER*4 LIB$SET_SCROLL,SYS$ASSIGN, BUFFER(512)
          CHARACTER*1 COLOR
          CHARACTER*80 STRING
          BYTE COLBYTE
          BYTE INPT,PASS
          EQUIVALENCE( COLOR, COLBYTE )
          COMMON /TRAP/INPT /CHAN/TT_CHAN
          CHARACTER*1 ANS,RET,PLT
C
C
          ANS = ' '
          COLOR = '3'
C
C set up a channel
C
          CALL SYS$ASSIGN('SYS$INPUT:',TT_CHAN,,)
 500      CONTINUE
          ISTART_LINE=23
          IEND_LINE=24
          TYPE *,'Enter P for phase, R for real, I for imaginary, M for modulus'
          READ(5,1000) PLT
          IF(PLT.NE.'P'.AND.PLT.NE.'R'.AND.PLT.NE.'I'.AND.PLT.NE.'M') GOTO 500
C
          XMAX = -1.E33
          XMIN = 1.E33
          IF(PLT.EQ.'P') THEN
            DO NI=1,SIZE
              IF(PHAS(NI).LT.XMIN) XMIN = PHAS(NI)
              IF(PHAS(NI).GT.XMAX) XMAX = PHAS(NI)
            END DO
            SCALE = (XMAX - XMIN)/(2**15-1)
            DO NI = 1,SIZE
              BUFFER(NI) = INT(PHAS(NI)/SCALE)
            END DO
          ELSE IF(PLT.EQ.'R') THEN
            DO NI = 1,SIZE
              IF(R(NI).LT.XMIN) XMIN = R(NI)
              IF(R(NI).GT.XMAX) XMAX = R(NI)
```

```
        END DO
        SCALE = (XMAX - XMIN)/(2**15-1)
        DO NI = 1,SIZE
          BUFFER(NI) = INT(R(NI)/SCALE)
        END DO
      ELSE IF(PLT.EQ.'M') THEN
        DO NI=1,SIZE
          IF(M(NI).LT.XMIN) XMIN = M(NI)
          IF(M(NI).GT.XMAX) XMAX = M(NI)
        END DO
        SCALE = (XMAX - XMIN)/(2**15-1)
        DO NI = 1,SIZE
          BUFFER(NI) = INT(M(NI)/SCALE)
        END DO
      ELSE
        DO NI = 1,SIZE
          IF(I(NI).LT.XMIN) XMIN = I(NI)
          IF(I(NI).GT.XMAX) XMAX = I(NI)
        END DO
        SCALE = (XMAX - XMIN)/(2**15-1)
        DO NI = 1,SIZE
          BUFFER(NI) = INT(I(NI)/SCALE)
        END DO
      END IF
      IF(ANS.NE.'O') THEN
C
C find the MIN MAX for the x and y coords
C
        MIN_Y = 2147483647
        MAX_Y = -2147483647
        DO NI = 1, SIZE
          IF(BUFFER(NI).LT.MIN_Y) MIN_Y = BUFFER(NI)
          IF(BUFFER(NI).GT.MAX_Y) MAX_Y = BUFFER(NI)
        END DO
C
C calculate where the axis ( zeroed ) should go
C
C x axis crossing
C
        IF(MIN_Y.LT.0.AND.MAX_Y.GT.0) THEN
          IXSTART = INT(FLOAT(MAX_Y)/FLOAT(ABS(MIN_Y)+MAX_Y) * 400.)
          IF(IXSTART.LE.0) IXSTART = 0
        ELSE IF(MAX_Y.LE.0) THEN
          IXSTART = 0
        ELSE IF(MIN_Y.GE.0.AND.MAX_Y.GE.0) THEN
          IXSTART = 400
        ELSE IF(MIN_Y.LE.0.AND.MAX_Y.LT.0) THEN
          IXSTART = 0
        END IF
C
C y axis crossing
C
        IYSTART = 0
      END IF
C
C plot the axis
C
      IF(ANS.EQ.'O') THEN
        CALL REENTER_REGIS
        IF(COLOR.LT.'3') THEN
          COLBYTE = COLBYTE + 1
        END IF
      END IF
C
```

```
      IF(ANS.EQ.'N'.OR.ANS.EQ.' ') THEN
        CALL ENTER_REGIS
        CALL FOREGROUND('3')
        STRING = 'MAX_Y '
        CALL NTEXT(STRING,MAX_Y, 10, 30)
        STRING = 'MIN_Y '
        CALL NTEXT(STRING,MIN_Y, 10, 50)
        CALL ABS_POSITION( 0, 0 )
        STRING = 'NUMBER OF POINTS '
        CALL NTEXT(STRING,SIZE, 10, 70)
      END IF
      CALL FOREGROUND('1')
      CALL ABS_POSITION( 0, IXSTART)
      CALL VEC_POINT( 799, 0 )
      CALL ABS_POSITION( IYSTART, 0 )
      CALL VEC_POINT( 0, 400 )
C
C now scale the points in the y-direction
C
      SCALE = FLOAT( MAX_Y - MIN_Y )/400.0
      IF(SCALE.EQ.0.0) SCALE = 1.0
      DO NI = 1, SIZE
        BUFFER(NI) = INT(FLOAT(BUFFER(NI)-MIN_Y)/SCALE)
      END DO
C
C after this the numbers should all be positive and, for the x points
C between 0 and 799, for the y points between 0 and 400...
C
C
C now plot that sucker
C
      CALL FOREGROUND(COLOR)
      X = 1
      Y = -1*(BUFFER(1) - 400)
      CALL ABS_POSITION( X, Y )
      CALL STRT_SEQ
      CALL VEC_POINT( 0, 0 )
      OLD_Y = Y
      OLD_X = X
      INCX = 1
      ISIZE = SIZE
      IF(SIZE.LT.390) THEN
        INCX = 2
        ISIZE = SIZE * 2
      END IF
      X = INCX
      DO NI = 1, SIZE
        Y = (-1*(BUFFER(NI) - 400) - OLD_Y)
        OLD_Y = -1*(BUFFER(NI) - 400)
        CALL POINT_ADD( X, Y )
      END DO
      CALL END_SEQ
      CALL ABS_POSITION( 0, 0 )
C
      CLOSE(2)
C
C ask for some info
C
      ISTAT = LIB$SET_SCROLL(ISTART_LINE, IEND_LINE)
      WRITE(6,1001)
      WRITE(6,1002)
 30   READ(5,1000) ANS
      IF(ANS.NE.'P'.AND.ANS.NE.'D') GOTO 30
      IF(ANS.EQ.'P') THEN
```

```
          ISTART_LINE = 1
          ISTAT = LIB$SET_SCROLL(ISTART_LINE, IEND_LINE)
          TYPE *,'BYE BYE'
          RETURN
        ELSE
          TYPE *,' O(verlay) file or N(ew dog)?'
          READ(5,1000) ANS
          GOTO 500
        END IF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCC
C ERRORS
CCCCCCCCCCCCCCCCCCCCCCCCCCC
C
 999    TYPE *,'Bad file.  Plot aborted...'
        RETURN
 1000   FORMAT(A)
 1001   FORMAT(' Enter a "P" to Piss off  ')
 1002   FORMAT(' or a "D" for a new Dog ',$)
        END
```

Appendix L

The raw image data reconstruction program RECO.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
        PROGRAM RECO
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
C *** MR2DVA ***
C
C Dr. Richard Hooper                    18-Dec-84
C Cross Cancer Institute
C Edmonton, Alberta, Canada
C
C A program to reconstruct a set of FID data from the Aspect-2000
C using a 2DFT technique, and create an output DELTA-11 file.
C
C--------------------------------------------------
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
C Modified by Steven Knudsen          3/04/86
C               10-102 Clinical Sciences Bldg
C
C This program simply ffts the raw time domain data and saves
C the resultant complex array(s) in a .RAW suffixed file.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
C
        IMPLICIT INTEGER*4 (I-N)
        LOGICAL*1 MATTYP,INNAM(60),ACCNAM(15),TMPNAM(60)
        CHARACTER*32 ANSWER
        CHARACTER*1 SAMPLE, ANS, ALREADY
        INTEGER*4 BUFF(128),N_BLOCK,M_BLOCK,NECHO
        COMMON /BB/ BUFF,N_BLOCK,M_BLOCK,SAMPLE,NECHO,ALREADY
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
        DATA  ACCNAM/'[','S','T','E','V','E','N','.','A','S',
        1     'P','E','C','T',']'/
        DATA ANS/'N'/
C--------------------------------------------------
        TYPE 100
100     FORMAT('0** ASPECT TO DELTA-11 2DFT RECONSTRUCTION ',
        1           'PROGRAM **')
C--------------------------------------------------
C Read file names
C--------------------------------------------------
1       TYPE 102
102     FORMAT(1X,/,'$Enter input file name:  ')
        DO 3 I=1,60
```

```
3       TMPNAM(I)=0
        ACCEPT 104,TMPNAM
104     FORMAT(60A1)
        IF(TMPNAM(1).EQ.' ') STOP
C
        IFLG=0
        DO 4 I=1,60
        IF(TMPNAM(I).EQ.'['.OR.TMPNAM(I).EQ.':') IFLG=-1
4       CONTINUE
        IF(IFLG.NE.0) THEN
          DO 5 I=1,60
5         INNAM(I)=TMPNAM(I)
        ELSE
          DO 7 I=1,15
7         INNAM(I)=ACCNAM(I)
          DO 9 I=16,60
          IM=I-15
9         INNAM(I)=TMPNAM(IM)
        END IF
        TYPE 111,INNAM
 111    FORMAT(' ',60A)
        DO 10 I=60,1,-1
        IF(INNAM(I).NE.' ') GO TO 14
10      INNAM(I)=0
C
C
14      TYPE 106
106     FORMAT(1X,/,'$Enter number of frequency points (64, ',
        1  '128 or 256): ')
        ACCEPT *,NFREQ
        IF(NFREQ.NE.64.AND.NFREQ.NE.128.AND.NFREQ.NE.256)
        1  GO TO 14
C
16      TYPE 107
107     FORMAT(1X,/,'$Enter number of phase points (64, ',
        1  '128 or 256): ')
        ACCEPT *,NPHASE
        IF(NPHASE.NE.64.AND.NPHASE.NE.128.AND.NPHASE.NE.256)
        1  GO TO 16
C
18      TYPE 108
108     FORMAT(1X,/,'$Enter number of echos(1, 4, 8, 16 ): ')
        ACCEPT *,NECHO
        IF(NECHO.LE.0.OR.NECHO.GT.16) GOTO 18
C
        IF(NPHASE.GT.NFREQ) GO TO 910
        NFLG=NFREQ
        IF(NFLG.EQ.64) THEN
          NBLOCK=16
          MATTYP='4'
        ELSE IF(NFLG.EQ.128) THEN
          NBLOCK=64
          MATTYP='6'
        ELSE
          NBLOCK=256
          MATTYP='8'
        END IF
C
C-----------------------------------------------------------
C
C Read input file,sort data from INTEGER*3 to INTEGER*4
C (ie. from Aspect to VAX/PDP format), and write to temporary output
C file (unit = 4).  Close input file (unit = 1).
C
```

```
C Then call the reconstruction routine. This routine fft's a..
C echos and stuffs the resultant arrays into a data file ( suffix
C .RAW ) on unit 7. Close unit 7.
C
C------------------------------------------------------------
 92     TYPE *,' ALREADY UNSCRAMBLED ( Y OR N )? '
        READ(5,1000) ALREADY
        IF(ALREADY.NE.'Y'.AND.ALREADY.NE.'N') GOTO 92
C
        IF(ALREADY.EQ.'N') THEN
C------------------------------------------------------------
        OPEN(UNIT=1,FILE=INNAM,ACCESS='DIRECT',
   *  STATUS='OLD',RECL=128,ASSOCIATEVARIABLE=N_BLOCK,ERR=900)
C------------------------------------------------------------
        END IF
C
        CALL UNSCRAM
        CALL FOURIER( INNAM )
        TYPE 118
118     FORMAT('0*** Reconstruction Completed! ***',/)
        GO TO 1
C------------------------------------------------------------
C Error routines
C------------------------------------------------------------
900     TYPE 902
902     FORMAT('0** ERROR ** Unable to open input image file')
        CLOSE (UNIT=1)
        STOP
C------------------------------------------------------------
910     TYPE 912
912     FORMAT('0** ERROR ** The number of frequency points '
        1     'must be greater than ',/,13x,
        2     'or equal to the number of phase points!',/)
        CLOSE (UNIT=1)
        STOP
 1000   FORMAT(A)
        END
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCCCCCCCCCCC
C
        SUBROUTINE UNSCRAM
C
C       THIS PROGRAM UNSCRAMBLES "FILE_NAME" (TRANSFFERED FROM ASPECT
C    USING "MSPR")
C       THE OUTPUT FILE WILL HAVE THE SAME NAME AS THE INPUT FILE AND
C       .DAT EXTENSION.
C       THE PROGRAM MUST BE RUN USING  $SYS$SYSDEVICE:UNSCRAM file_name.ext
C
C
C       IN_BYTES        1  2  3 ...4  5  6 ..........
C                       | | |  | | |
C                       V  V  V  | V  V
C       OUT_BYTES       1  2  3  4| 5  6  7  8 ...
C                                 ^ |__________^  ^
C                                 |_____sign_____|
C
C       VER #1   MAY,30 1984      U&A, APPLIED SCIENCES IN MEDICINE
C                                 DAN GHEORGHIU
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
```

```
C S Knudsen             28-May-86
C
C Modified for use in 2dfft reconstruction routine 'phas2dap'.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C S Knudsen             8-Jul-86
C
C Modified to use the array processor
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        IMPLICIT INTEGER*4 (A-Z)
C
        BYTE            IN_BUFFER(512)
        BYTE            STACK(4),TEMP
        BYTE            LONG_BUFFER(1024)
        INTEGER*4       OUT_BUFFER4(128),IN_BUFFER4(128),BUFF(128)
        INTEGER*4       IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE, F_SIZE
C
        INTEGER*4 ACQ_FLAG, ACQ_MASK, NECHO
        LOGICAL*1 MATTYP, SAMP_MODE
        CHARACTER*1 YES, SAMPLE,ALREADY,ANSWER
C
        COMMON /BB/ BUFF,N_BLK,M_BLK,SAMPLE,NECHO,ALREADY
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
        COMMON /MODE/ IMODE, ICODE, MSP_ERR
        COMMON /UNSC/           IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
C
        EQUIVALENCE     (STACK,STACK4)
        EQUIVALENCE     (IN_BUFFER,IN_BUFFER4)
        EQUIVALENCE     (LONG_BUFFER,F_SIZE)
C---------------------------------------------
        BYTE            STRING(4)/27,'[','6','n'/
        BYTE            BELL/7/
        BYTE            REPORT(20)
        INTEGER*2       IOSB(4)
        INTEGER*4       SYS$ASSIGN,SYS$QIO,SYS$WAITFR
        INTEGER*4       TERMIN_MASK,FUNC_CODE
C
        INCLUDE                 '($IODEF)'
        INCLUDE                 '($SSDEF)'
C---------------------------------------------
        DATA    IMODE/3/
        DATA ACQ_MASK/'000000C0'X/
        SAMP_MODE = .TRUE.
        IN_CHANNEL = 1
        OUT_CHANNEL = 4
C
C------------------------------------------------------
C
C check to see if this has already been done
C
        IF(ALREADY.NE.'Y'.AND.ALREADY.NE.'N') THEN
          TYPE *,'Oops,  I do not remember being unscrambled or do I ? ...'
          TYPE *,' bye...'
          STOP
        END IF
C
C
        OPEN(UNIT=4,FILE='[STEVEN.ASPECT]DUMDUM.TP1',
        1          ACCESS='DIRECT',STATUS='OLD',RECL=128,
     2       ASSOCIATEVARIABLE=M_BLK,ERR=930)
101  CONTINUE
     IF(ALREADY.EQ.'Y') THEN
       TYPE *,' Sample mode ( M or Q )'
       READ(5,1000) SAMPLE
       IF(SAMPLE.NE.'M'.AND.SAMPLE.NE.'Q') GOTO 101
       CLOSE(UNIT=1)
       RETURN
     END IF
C-------------------------------------------------------------
C
C check out the first block for aquisition parameters, namely SEQ or SIM
C sampling
C
C
C    GET THE FILE_SIZE IN BYTES AND TRANSFER HEADER BLOCK AS IT IS.
C    - 1 LONG_WORD = FILE_SIZS/3
C    - 648 BYTES (216 ASPECT_WORDS) USER INFO
C
     N_BLK=1                          ! BLOCK VARIABLE OF INPUT BUFFER
     M_BLK=1                          ! BLOCK VARIABLE OF OUTPUT BUFFER
C
     READ(1'N_BLK) (LONG_BUFFER(I),I=1,512)
     READ(1'N_BLK) (LONG_BUFFER(I),I=513,1024)
     FILE_SIZE=F_SIZE*3
C
C
     OUT_128=1                        ! OUTPUT BUFFER COUNTER
     OUT_BUFFER4(1)=FILE_SIZE
     L=0                              ! BYTE COUNTER OF DATUS
     J=0                              ! POINTS EVEN DATUS
C
D    TYPE *,'FILE_SIZE=',FILE_SIZE
C
     DO 110 N_BYTES=5,664
C
         L=1+JMOD(L,3)                ! INCREMENT BYTE PTR OF DATUS
         STACK(L)=LONG_BUFFER(N_BYTES)        ! COPY UP TO 3 BYTES
         IF(L.EQ.3) THEN
             J=1+JMOD(J,2)            ! INCREMENT EVEN PTR OF DATA
             IF(J.EQ.2) THEN          ! EVEN DATUS NEEDS SWAPING
                 TEMP=STACK(1) ! PERMUTE 3 BYTE DATUS
                 STACK(1)=STACK(2)
                 STACK(2)=STACK(3)
                 STACK(3)=TEMP
             ENDIF
C
             STACK(4)=0
C
C            CHECK IF OUTPUT_BUFFER IS FULL
C
             OUT_128=1+JMOD(OUT_128,128)  ! OUTPUT COUNTER INCREMENT
             OUT_BUFFER4(OUT_128)=STACK4
             IF(OUT_128.EQ.128) THEN
D                TYPE *,' M_BLK=  ',M_BLK
                 WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
                 IF(SAMP_MODE) THEN
                   ACQ_FLAG = ACQ_MASK.AND.OUT_BUFFER4(45)
                   SAMP_MODE = .FALSE.
                 END IF
             ENDIF
C
         ENDIF
C
```

```
110     CONTINUE
C
        IF(OUT_128 .NE. 128) THEN
                OUT_128 = OUT_128 +1
                DO J=OUT_128 ,128
                        OUT_BUFFER4(J)=0
                ENDDO
                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
        ENDIF
C
C find out what type of sampling was used
C
        IF(ACQ_FLAG.EQ.'000000C0'X) THEN
          TYPE *,'The data is sampled simultaneously'
          SAMPLE = 'M'
        ELSE
          TYPE *,'The data is sampled sequentially'
          SAMPLE = 'Q'
        END IF
C
C       Call MSP, if not busy !
C
        CALL MSATT_UNS(0,ISTATUS)     !copy after Fipak's MSATT
C                                                 (returns MSP status)
D       TYPE *,'ISTATUS=',ISTATUS
        IF(ISTATUS .EQ. SS$_NORMAL) THEN
                ASP_FILE_SIZE = FILE_SIZE/3
                CALL UNSCRAM_WITH_MSP
                IF(MSP_ERR .NE. 1) GOTO 80
        ENDIF
C
C       MSP busy, let's do it on VAX
C
C
C       CONVERT THE REST OF DATA
C       #ASP_WORDS=#VAX_LONG_WORDS
C
        IN_512=0                                  ! INPUT BUFFER COUNTER
        OUT_128=0                                 ! OUTPUT BUFFER COUNTER
        L=0                                       ! BYTE COUNTER OF DATUS
        J=0                                       ! POINTS EVEN DATUS
C
        DO 10 N_BYTES=1,FILE_SIZE
C
                IN_512=1+JMOD(IN_512,512)
                IF(IN_512.EQ.1) READ(1'N_BLK) (IN_BUFFER4(I),I=1,128)
D       TYPE *,'N_BLK=',N_BLK
                L=1+JMOD(L,3)                     ! INCREMENT BYTE PTR OF DATUS
                STACK(L)=IN_BUFFER(IN_512)        ! COPY UP TO 3 BYTES
                IF(L.EQ.3) THEN
                        J=1+JMOD(J,2)             ! INCREMENT EVEN PTR OF DATA
                        IF(J.EQ.2) THEN           ! EVEN DATUS NEEDS SWAPING
                                TEMP=STACK(1)     ! PERMUTE 3 BYTE DATUS
                                STACK(1)=STACK(2)
                                STACK(2)=STACK(3)
                                STACK(3)=TEMP
                        ENDIF
C
                        IF(STACK(3).LT.0) THEN    ! TAKE CARE OF SIGN
                                STACK(4)=-1
                        ELSE
                                STACK(4)=0
                        ENDIF
C
```

```
C                       CHECK IF OUTPUT_BUFFER IS FULL
C
                        OUT_128=1+JMOD(OUT_128,128)  ! OUTPUT COUNTER INCREMENT
                        OUT_BUFFER4(OUT_128)=STACK4
                        IF(OUT_128.EQ.128) THEN
D                           TYPE *,' M_BLK=  ',M_BLK
                            WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
                        ENDIF
C
                ENDIF
C
10      CONTINUE
C
C       WRITE THE LAST BLOCK IF ANY
C
130     IF(OUT_128.NE.128) THEN
D               TYPE *,' LAST M_BLK=  ',M_BLK
                WRITE(4'M_BLK)(OUT_BUFFER4(I),I=1,128)
        ENDIF
C
C       NOW CLOSE ALL FILES AND EXIT.
C
80      CLOSE(UNIT=1)
C
        TYPE *,' OUT OF UNSCRAM'
        RETURN
 1000   FORMAT(A)
C
C-----------------------------------------------------------
C Error routines
C-----------------------------------------------------------
930     CONTINUE
        OPEN(UNIT=4,FILE='[STEVEN.ASPECT]DUMDUM.TP1',
        1       ACCESS='DIRECT',STATUS='NEW',RECL=128,
        2       ASSOCIATEVARIABLE=M_BLK,ERR=931)
        GOTO 101
931     TYPE 932
932     FORMAT('0** ERROR ** Unable to open file DUMDUM.TP1')
        GOTO 999
C
950     TYPE 952
952     FORMAT('0** ERROR ** Error reading input image file')
        GOTO 999
985     TYPE 987
987     FORMAT('0** ERROR ** Error writing file DUMDUM.TP1')
        GOTO 999
C
999     CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        CLOSE (UNIT=4)
        STOP
        END
C
C*****************************************************************
C
        SUBROUTINE     UNSCRAM_WITH_MSP
C
C       Subroutine assumes that the header block has been copied !
C     For more detail see MSP array processor manual and/or talk
C       with system manager
C
C       Dan G.
C--------------------------------------------------------------
```

```
C
        INTEGER*4       IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
        INTEGER*4       MSP_REG(16)
        INTEGER*4       MSP_DRAM       /' 400000'X/
        INTEGER*4       MSP_PROGRAM /' C01400'X/
        INTEGER*4       MSP_START      /' C0140C'X/
        INTEGER*4       BUFFER(128)
        INTEGER*4       ICHANNEL
C
        COMMON /MODE/           IMODE,ICODE,MSP_ERR
        COMMON /UNSC/           IN_CHANNEL,OUT_CHANNEL,ASP_FILE_SIZE
C
D       TYPE *,'IMODE ',IMODE
D       IMODE_OLD=IMODE
C
        CALL MSCLR                              !CLEAR MSP
        IF(IWAIT('MSCLR ').NE.0)GO TO 92
        CALL ENBDMA                             !ENABLE DMA
C
C       Load mini_program
C
        ICHANNEL=3
        CALL ASSIGN(ICHANNEL,'DISK$USER:[STEVEN.ASPECT]UNSCMISAL.EXE')
        NBLKS=3
        NBYTES=512
        CALL VMNLOAD(ICHANNEL,MSP_PROGRAM,NBLKS,NBYTES)         !LOAD MINI
PROGRAM
        IF(IWAIT('MNLOAD').NE.0)GO TO 92
        CALL CLOSE(ICHANNEL)
C
        CALL PSWBSH('1'X)                       !Disable FPP interupt
        IF(IWAIT('PSWBSH').NE.0)GO TO 92
C
        N_BLK = 3
        M_BLK = 3
        N_DATA= 0                               !general data_counter
        N_MAX_MSP='20000'X                      !MSP_memory capacity
        MSP_REG(11) = MSP_DRAM                          !R2 <= unsc_data_address
        MSP_REG(16) = MSP_DRAM + '20000'X       !R2 <= sc_data_address
C
        DO WHILE( N_DATA .LT. ASP_FILE_SIZE )
                N_DATA_MSP = ASP_FILE_SIZE - N_DATA
                IF( N_DATA_MSP .GT. N_MAX_MSP) N_DATA_MSP = N_MAX_MSP
C
C               Read input file and put data into MSP
C
                N_LOOP= 0
                N_MSP = 0
                MSP_ADDR= MSP_DRAM + '20000'X           !start loading here
D               TYPE 85,MSP_ADDR
D               IMODE=3
85              FORMAT(' ',4Z12)
c
                DO WHILE(N_MSP .LT. N_DATA_MSP)
                        READ (IN_CHANNEL'N_BLK,ERR=100) BUFFER
                        CALL    VPUT(BUFFER,MSP_ADDR,4,128)
                        MSP_ADDR= MSP_ADDR +512
                        N_LOOP = N_LOOP + 1
                        N_MSP = (512*N_LOOP)/3
CD                      TYPE 85,MSP_ADDR,N_BLK,N_LOOP,N_MSP
                        N_BLK = N_BLK + 1
                        IF      (IWAIT('BUFFER').NE.0)  GO TO 92
                ENDDO
D               TYPE 85,MSP_ADDR
```

```
D               IMODE=IMODE_OLD
C
C               Load MSP registers with "unscrambling" parameters
c
C
                CALL MSCLR                              !CLEAR MSP
                IF(IWAIT('MSCLR ').NE.0)GO TO 92
                CALL PSWBSH('1'X)                               !Disable FPP interupt
                IF(IWAIT('PSWBSH').NE.0)GO TO 92
C
                MSP_REG(13) = N_DATA_MSP                        !R4 <= counter
D               TYPE 85,MSP_REG(11),MSP_REG(13),MSP_REG(16)
C
                CALL    MSWRTE('9400'X,MSP_REG)                                 !R2,R4,R7
                IF      (IWAIT('WR9400').NE.0)GO TO 92
D2000   CONTINUE        !after break
C
                CALL    VMNSTRT ( MSP_START )
                IF      (IWAIT('STUNSC').NE.0)GO TO 92
C
C               Retrieve unscrambled_data from MSP into output_file
C
                M_MSP = 0
                MSP_ADDR= MSP_DRAM
D               TYPE 85,MSP_ADDR
D               IMODE=3
                DO WHILE(M_MSP .LT. N_DATA_MSP)
                        CALL    VGET(BUFFER,MSP_ADDR,4,128)
                        M_MSP = M_MSP + 128
                        IF(IWAIT('BUFFER').NE.0) GO TO 92
C
                        IF(M_MSP .GT. N_DATA_MSP) THEN ! zero_filling in last
                                M_FILL_ZERO=1+MOD(N_DATA_MSP,128)
                                DO JJ=M_FILL_ZERO,128  ! block ,following data.
                                        BUFFER(JJ)=0
                                ENDDO
                        ENDIF
C
                        WRITE (OUT_CHANNEL'M_BLK,ERR=110) BUFFER
                        M_BLK = M_BLK +1
                        MSP_ADDR= MSP_ADDR +512
                ENDDO
D               TYPE 85,MSP_ADDR
D               IMODE=IMODE_OLD
C
                N_DATA = N_DATA + N_DATA_MSP
        ENDDO ! N_DATA loop
C
92      CALL MSDET
        RETURN
C
100     TYPE *,'*** ERROR  reading input_file '
        GOTO 92
C
110     TYPE *,'*** ERROR  writing output_file '
        GOTO 92
C
        END
C
C*********************************************************************
        SUBROUTINE MSATT_UNS (LUN,ISTATUS)
C
C
C       Copyright (c) 1982,1983
```

```
C
C     Computer Design and Applications, Inc.,
C     411 Waverly Oaks Road
C     Waltham, Massachusetts 02154
C
C     All rights reserved.
C
C
C     MSATT - Attach MSP-3000 to task
C
C     This subroutine attaches the MSP-3000 to the task for
C     exclusive use. No other user may attach it while it is
C     in use.
C
C
C     Version 0.0           15-Feb-83      Peter Kapinos
C
C                           First Release
C
C
      IMPLICIT              NONE
C
      INCLUDE                      '($SSDEF)'
      INCLUDE                      '($IODEF)'
      INTEGER*4             LUN,ISTATUS
      INTEGER*4             SERVICE_STATUS
      INTEGER*4             SYS$ASSIGN
      INTEGER*4             SYS$SETEF
      INTEGER*4             FINAL_IO_STATUS
      INTEGER*4             SYS$QIOW
      INTEGER*4             IO$_ALTDMA
      INTEGER*4             BITS_TO_SET
      INTEGER*4             BITS_TO_CLEAR
      CHARACTER*5           M3K_NAME
      LOGICAL*1             FPP_SAFE
      INTEGER*2             IOSB (4)
      INTEGER*4             EVENT_FLAG,M3K_CHANNEL
C
      COMMON      /QIO_ARGUMENTS/      EVENT_FLAG,M3K_CHANNEL,IOSB
      COMMON      /SAFE_MODE/   FPP_SAFE
C
C     Establish the MSP-3000 device name for the assign
C
      PARAMETER             (M3K_NAME  = 'CDA0:')
C
C     Equate MSP-3000 $QIO function code with existing DEC function code
C
      PARAMETER             (IO$_ALTDMA = IO$_ACCESS)
      PARAMETER             (BITS_TO_SET = 0)
      PARAMETER             (BITS_TO_CLEAR = 'FFFFFFFF'X)
C
C     Start of executable code
C     ------------------------
C
C     Establish the event flag number for all system services
C
      EVENT_FLAG = 32
C
C     Default mode at attach is safe
C
      FPP_SAFE = .TRUE.
C
C     Set the event flag so the first MSWAIT works correctly
C
```

```
        SERVICE_STATUS         = SYS$SETEF (%VAL (EVENT_FLAG))
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .EQ. SS$_WASSET) GO TO 400
C
        IF (SERVICE_STATUS .EQ. SS$_WASCLR) GO TO 400
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,310) SERVICE_STATUS
310       FORMAT (' MSATT SETEF Failed with code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (SERVICE_STATUS)
C
                ISTATUS=SERVICE_STATUS
                RETURN
        END IF
C
C       Assign the MSP-3000 to this task
C
400     SERVICE_STATUS         = SYS$ASSIGN ( M3K_NAME,M3K_CHANNEL,,)
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,410) SERVICE_STATUS
410       FORMAT (' MSATT Assign Failed with code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (SERVICE_STATUS)
C
                ISTATUS=SERVICE_STATUS
                RETURN
        END IF
C
C       Issue QIO to clear the ALTDMA mask stored in the UCB
C
        SERVICE_STATUS         = SYS$QIOW ( %VAL (EVENT_FLAG), %VAL (M3K_CHANNEL),
     •                         %VAL (IO$_ALTDMA),IOSB,,,
     •            %VAL (BITS_TO_SET),
     •                                 %VAL (BITS_TO_CLEAR),,,,)
C
C       Error handling Section
C
C       Check SERVICE_STATUS to see if call and arguments were accepted by VMS.
C
        IF (SERVICE_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,510) SERVICE_STATUS
```

```
510       FORMAT (' MSATT $QIO Failed with code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (SERVICE_STATUS)
C
                ISTATUS=SERVICE_STATUS
                RETURN
        END IF
C
C       Check the I/O Status Block to see if there were any I/O errors
C
        FINAL_IO_STATUS = IOSB (1)
C
        IF (FINAL_IO_STATUS .NE. SS$_NORMAL) THEN
C
          CALL BLANK (1)
C
          WRITE (5,600) FINAL_IO_STATUS
600       FORMAT (' MSATT I/O Failed With Code = ',I9)
C
C         This error is serious so exit from image
C
C         CALL EXIT (FINAL_IO_STATUS)
C
                ISTATUS=SERVICE_STATUS
                RETURN
        END IF
C
C       Put a success code in I/O Status Block so first MSWAIT works
C       correctly
C
        IOSB (1) = SS$_NORMAL
        ISTATUS  = SS$_NORMAL
C
C       We're done here so return.
C
        RETURN
C
        END

C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
        SUBROUTINE FOURIER( ISTRING )
C
C Read temporary file (unit = 4) into virtual array, and filter
C if necessary.  Calculate 2dft of virtual array, normalize
C and save the complex data set(s).
C Finally, close all files.
C
        IMPLICIT INTEGER*4 (I-N)
        LOGICAL*1 MATTYP,ISTRING(60)
        INTEGER*2 BUFFW(256),IWDUM, NSHIFT(32), NMULT(32)
        INTEGER*4 BUFFI(128),APBASE,APADDR,ATAB1,ATAB2,ATAB3,ATAB4,
        1         NECHO,BUFF(128)
        REAL*4 BUFFR(128)
        COMPLEX*8 BUFFC(64),ZMAT(256,256),CDUM,Z2MAT(256,256)
        COMPLEX ZTEMPC(256), ZTEMP
        REAL ZTEMPR(512)
        BYTE RAWDATA(60)
        CHARACTER*32 ANSWER
        CHARACTER*1 SAMPLE, SHIF, ALREADY
        CHARACTER*60 RAWSTRING
C
C
        EQUIVALENCE (RAWSTRING,RAWDATA(1))
        EQUIVALENCE (ZTEMPC(1),ZTEMPR(1))
        EQUIVALENCE (BUFFI(1),BUFFW(1),BUFFR(1),BUFFC(1))
C
C
        COMMON /BB/ BUFF,N_BLOCK,M_BLOCK,SAMPLE,NECHO,ALREADY
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
C
C copy the parameter passed
C
        DO I = 1, 60
          RAWDATA(I) = ISTRING(I)
        END DO
C
C
C Open a file to contain the complex arrays; 1 array per echo
C
        DO I = 1,60
          IF(RAWDATA(I).EQ.'.') IPERIOD = I
        END DO
        RAWDATA(IPERIOD+1) = 'R'
        RAWDATA(IPERIOD+2) = 'A'
        RAWDATA(IPERIOD+3) = 'W'
        TYPE *,' IN FOURIER. RAWDATA = ',RAWSTRING
        OPEN( UNIT=7,FILE=RAWDATA,FORM='UNFORMATTED',STATUS='NEW',ERR=980 )
C
C if requested open the file containing the number of points/echo to
C shift the echo in time ( refer to PHASCOR.FOR )
C
 1      CONTINUE
        TYPE *,'Open the file containing shift and phase corrections',
        1       ' each echo ( Y or N )'
        READ(5,1000) SHIF
        IF(SHIF.NE.'Y'.AND.SHIF.NE.'N') GOTO 1
        IF(SHIF.EQ.'Y') THEN
          OPEN(UNIT=8,FILE='[STEVEN.ASPECT]SHIFTS.REC',STATUS='OLD',
        1       FORM='UNFORMATTED',ERR=931)
          READ(8) NSHIFT,NMULT
          CLOSE(UNIT=8)
        END IF
C
C process NECHO echos from the dumdum file
C
        DO NE = 1, NECHO
C-----------------------------------------------------------
C Zero virtual array
C-----------------------------------------------------------
          DO I = 1, 512
          ZTEMPR(I) = 0.0
          END DO
C
          DO 12 J=1,NFLG
          DO 10 I=1,NFLG
10        ZMAT(I,J)=CMPLX(0.0,0.0)
12        CONTINUE
C-----------------------------------------------------------
C Read file into virtual array. Remember that the echos are
C Note that if there is zero filling, the zeros are placed at
C the beginning and end of the matrix.
C-----------------------------------------------------------
C
```

```
          NFR=NFREQ/64
          NPHHA=NPHASE/2
          NFRHA=NFREQ/2
          NFLHA=NFLG/2
          M_BLOCK= ( NE - 1 )*NFR + 3
          M_BSTEP = NFR * NECHO
C
          DO 20 JJ=1,NPHASE
          M_BLOCKOLD = M_BLOCK
          IF(NPHASE.EQ.NFLG) THEN
            JPH=JJ
          ELSE
            JPH=JJ+NFLHA-NPHHA
          END IF
C
          DO 18 KK=1,NFR
            READ (4'M_BLOCK,ERR=980) BUFFI
C
            DO 14 I=1,128
14            BUFFR(I)=REAL(BUFFI(I))        ! Convert from INT*4 to REAL*4
C
            DO 16 I=1,64                     ! Stuff freq data into cols
              IFR=((KK-1)*64)+I              ! starting at 1 + NFLHA - NPHHA
              ZMAT(IFR,JPH)=BUFFC(I)
16          CONTINUE
18        CONTINUE
          M_BLOCK = M_BLOCKOLD + M_BSTEP
20        CONTINUE
C
C Now, if the number of shifts for the time data / echo has been read in
C then shift the frequency time data by the indicated shift for that echo
C
        IF(SHIF.EQ.'Y') THEN
C------------------------------------------------------------
C
C At this point we have the echo data in the array ZMAT
C
C Now let's look at the middle + 1 scan in that image.
C
C copy it into the real and imaginary arrays used in the FFT
C
C for every loop this data array may need to be shifted n points
C right or left. This shift need be a logical shift
C
C         1  2  3  4 ...                    NFLG-1 NFLG
C i.e     +--+--+--+--+-            -+--+--+
C         | | | | | ...     --| | |         right logical shift
C         +--+--+--+--+-            -+--+--+
C       ^ -> -> ->                            |
C          /\                                 |
C          |                                  |
C          +-------------------------------+
C
C The variable ISHIFT indicates the direction and number of points to
C shift the time data
C                    ISHIFT = a negative integer then shift left
C                           = a positive  "    "    " right
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C
C get the number of points to shift left ( -ve integer ) or right
C ( +ve integer )
C
```

```
        ISHIFT = NSHIFT(NE)
C
C shift the points as indicated by ISHIFT
C
         IF(ISHIFT.NE.0) THEN
          IF(ISHIFT.LT.0) THEN                 !shift left
           DO I = 1,-1*ISHIFT
                DO JJ=1,NPHASE
                 IF(NPHASE.EQ.NFLG) THEN
                  JPH=JJ
                 ELSE
                     JPH=JJ+NFLHA-NPHHA
                 END IF
                 ZTEMP = ZMAT(1,JPH)
                 DO II = 1,NFLG-1
                     ZMAT(II,JPH) = ZMAT(II+1,JPH)
                 END DO
                 ZMAT(NFLG,JPH) = ZTEMP
            END DO
               END DO
          ELSE                                  !shift right
           DO I = 1,ISHIFT
                DO JJ=1,NPHASE
                 IF(NPHASE.EQ.NFLG) THEN
                  JPH=JJ
                 ELSE
                     JPH=JJ+NFLHA-NPHHA
                 END IF
                 ZTEMP = ZMAT(NFLG,JPH)
                 DO II = NFLG,2,-1
                     ZMAT(II,JPH) = ZMAT(II-1,JPH)
                 END DO
                 ZMAT(1,JPH) = ZTEMP
            END DO
           END DO
          END IF
         END IF
        END IF
C-----------------------------------------------------------
C
C put the columns into the MSP
C
        CALL MSATT(0,0)
        CALL MSCLR
        CALL ENBDMA
        APBASE='400000'X
        NFLG8=8*NFLG
        DO J=1,NFLG
          APADDR=APBASE+(J-1)*NFLG8
          CALL WAIT(0)
          DO II = 1, 256
            ZTEMPC(II) = ZMAT(II,J)
          END DO
          CALL VPUT(ZTEMPR(1),APADDR,4,NFLG*2)
        END DO
C
C check the sampling technique
C
        IF(SAMPLE.EQ.'Q') THEN
C
C negate every other complex point in the array
C
          INDEX = 8 * 2
          NPOINTS = NFREQ*NFREQ/2
          APADDR = APBASE + 8
          CALL CVNEG( APADDR, INDEX, APADDR, INDEX, NPOINTS )
C
C and do a real fold about the x axis
C
          INCAD = 4              ! INCREMENT PER ROW (BYTES/POINT)
          NPNTS = NFREQ * 2      ! NUMBER OF REAL POINTS
          ATAB1 = APBASE
          ATAB2 = APBASE + INCAD*(NPNTS/2 - 1)
          ATAB3 = APBASE + INCAD* NPNTS/2
          ATAB4 = APBASE + INCAD*(NPNTS  - 1)
          NQUFR = NPNTS/4
          NCNT12 = NFREQ
          NCNT34 = NFREQ
          INDEX = NFREQ * 8
C
          DO I = 1, NQUFR
            CALL VSWAP(ATAB1,INDEX,ATAB2,INDEX,NCNT12)
            CALL VSWAP(ATAB3,INDEX,ATAB4,INDEX,NCNT34)
C
            ATAB1 = ATAB1 + INCAD
            ATAB2 = ATAB2 - INCAD
            ATAB3 = ATAB3 + INCAD
            ATAB4 = ATAB4 - INCAD
          END DO
        END IF
C-----------------------------------------------------------
C Perform 2dft
C-----------------------------------------------------------
        CALL WAIT(1)
        IF(SAMPLE.EQ.'M') THEN
C
C do a 2d complex fft
C
          CALL C2FFT(APBASE,8,NFLG,+1)
        ELSE
C
C do a real fft by columns
C
          NFLG8 = NFLG * 8
          DO J = 1,NFLG
            APADDR=APBASE+(J-1)*NFLG8
            CALL RFFT( APADDR, 8, NFLG, +1 )
          END DO
C
C complex fft it by rows
C
          INDEX = 8
          INDX1 = NFREQ * INDEX
          INCAD = INDEX
          DO J = 1,NFLG
            APADDR = APBASE + (J-1)*INCAD
            CALL CFFT( APADDR,INDX1, NFLG, +1 )
          END DO
        END IF
C
C do corrections here
C
CCC
C
C if sequential sampling correct for sign change here
C
        IF(SAMPLE.EQ.'Q') THEN
C
```

```
C multiply every other column by -1
C
            CALL WAIT(2)
            INDEX = NFREQ*8 * 2
            NPOINTS = NFREQ/2
            DO J = 1,NFREQ
              APADDR = APBASE + (J-1)*8
              CALL CVNEG( APADDR, INDEX, APADDR, INDEX, NPOINTS )
            END DO
          END IF
C
C reorder the phase terms
C
          CALL WAIT(2)
          NCNT12 = NFREQ*2
          ATAB1 = APBASE
          ATAB2 = APBASE + NFLG8*NFREQ/2
          INDEX = 4
          DO J = 1,NFREQ/2
            CALL VSWAP(ATAB1,INDEX,ATAB2,INDEX,NCNT12)
            ATAB1 = ATAB1 + NFLG8
            ATAB2 = ATAB2 + NFLG8
          END DO
C
C check for the change of sign due to echo number ( 180 degree inversion )
C
          IF(SHIF.EQ.'Y') THEN
            IF(NMULT(NE).GT.0) THEN
C
C negate every point
C
              CALL WAIT(2)
              INDEX = 8
              NPOINTS = NFREQ*NFREQ
              APADDR = APBASE
              CALL CVNEG( APADDR, INDEX, APADDR, INDEX, NPOINTS )
            END IF
          END IF
C------------------------------------------------------------
C Copy array from array processor
C------------------------------------------------------------
C
          CALL WAIT(2)
          DO 32 J=1,NFLG
            APADDR=APBASE+(J-1)*NFLG8
            CALL WAIT(2)
            CALL VGET(ZTEMPR(1),APADDR,4,NFLG*2)
            DO II = 1, 256
              ZMAT(J,II) = ZTEMPC(II)
            END DO
32        CONTINUE
C------------------------------------------------------------
C Detach from array processor
C------------------------------------------------------------
          CALL WAIT(3)
          CALL MSDET
C------------------------------------------------------------
C Reorder frequency and phase terms
C------------------------------------------------------------
          IF(SAMPLE.EQ.'M') THEN
            DO 36 J=1,NFLG
            DO 34 I=1,NFLHA                  ! Reorder frequency terms
            IP=I+NFLHA
            CDUM=ZMAT(J,I)
```

```
               ZMAT(J,I)=ZMAT(J,IP)
               ZMAT(J,IP)=CDUM
34             CONTINUE
C
               DO 35 I=1,NFLHA                       ! and flip about x axis
               IP=NFLG+1-I
               CDUM=ZMAT(J,I)
               ZMAT(J,I)=ZMAT(J,IP)
               ZMAT(J,IP)=CDUM
35             CONTINUE
36             CONTINUE
             END IF
C-----------------------------------------------------------------
C
C save this echo on disk ( unformatted )
C
        WRITE(7) ZMAT
C-----------------------------------------------------------------
C-----------------------------------------------------------------
        ENDDO
        CLOSE (UNIT=4)
        CLOSE ( UNIT=7 )
C
        RETURN
1000    FORMAT(A)
C
C ERRORS
C
980     CONTINUE
        TYPE *,' Error opening raw data file. Bye bye.....'
        STOP
931     CONTINUE
        TYPE *,' Error opening shifts file. Bye bye.....'
        STOP
        END

C
C*****************************************************************
C
C       WAIT.FTN
C
C       17-AUG-83
C       MARTIN HOERRMANN
C       BRUKER MEDIZINTECHNIK GMBH
C       7512 RHEINSTETTEN-F., D.
C
C       SUBROUTINE WAIT(III) to type error-messages if MSP works bad
C
C*****************************************************************
C
C       EDIT #1         PDP -> VAX              11-MAY-84
C
C       Dr. Richard Hooper
C       Cross Cancer Institute
C       Edmonton, Alberta, CANADA
C
C*****************************************************************
C
        SUBROUTINE WAIT(III)
C
        INTEGER*4 IERARR(9)
C
        CALL MSWAIT(IERR,ICODE)
        IF(IERR.NE.1) RETURN    !NO ERROR IF 1
        IF(ICODE.NE.0.AND.ICODE.NE.5.AND.ICODE.NE.6.AND.ICODE.NE.7)
        1         GOTO 20
        CALL ERRINF(ICODE,IERARR)      !Get error-information
        WRITE(6,10) ICODE,(IERARR(J),J=1,9)
10      FORMAT(' ***  MSP-ERROR ***'/
        1         ' IERR = ',I5/
        2         ' NAN-C =',O14/
        3         ' U-C   =',O14/
        4         ' O-C   =',O14/
        5         ' I-C      =',O14/
        6         ' NAN-T =',O14/
        7         ' U-T   =',O14/
        8         ' O-T   =',O14/
        9         ' I-T   =',O14/
        1         ' ERRT  =',O14)
C                                      !Return too, if FPP-error
20—     WRITE(6,102) III,IERR,ICODE        !PRINT MSP ERROR CODE
102     FORMAT(' WAIT> I,IERR,CODE=',3I7)
        IF(ICODE.EQ.0) WRITE(6,103)
103     FORMAT(' bad.data?')
        IF(ICODE.EQ.5) WRITE(6,105)
105     FORMAT(' Illegal MSP-memory access?')
        IF(ICODE.EQ.6) WRITE(6,106)
106   FORMAT(' MSP-unibus error')
        IF(ICODE.EQ.7) WRITE(6,107)
107     FORMAT(' MSP overheated?')
        RETURN
        END
```

Appendix M

The image generation program IMGMAKE.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        PROGRAM IMGMAKE
C
C               Steven Knudsen  20-May-85
C               10-102 Clinical Sciences Bldg
C
C       Original source: Rick Hooper, W W Cross Cancer Institute
C
C This program allows for the production of real, imaginary, phase
C and modulus images depending on the parameter chosen at run time.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        IMPLICIT INTEGER*4 (I-N)
        LOGICAL*1 MATTYP,INNAM(60),ACCNAM(15),TMPNAM(60)
        CHARACTER*32 ANSWER
        CHARACTER*1 SAMPLE, ANS, PREPROC
        INTEGER*4 BUFF(128),N_BLOCK,M_BLOCK,NECHO
        COMMON /BB/ BUFF,N_BLOCK,M_BLOCK,SAMPLE,NECHO
        COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
C
C set up default file for raw data to be in
C
        DATA  ACCNAM/'[','S','T','E','V','E','N','.','A','S',
        1    'P','E','C','T',']'/
        DATA ANS/'N'/
C
        TYPE 100
100     FORMAT('0** IMAGE CONSTRUCTION FROM RAW DATA FILE **')
C
C Read file names
C
1       TYPE 102
102     FORMAT(1X,/,'$Enter input file name:  ')
        DO 3 I=1,60
3       TMPNAM(I)=0
        ACCEPT 104,TMPNAM
104     FORMAT(60A1)
        IF(TMPNAM(1).EQ.' ') STOP
C
        IFLG=0
        DO 4 I=1,60
        IF(TMPNAM(I).EQ.'['.OR.TMPNAM(I).EQ.':') IFLG=-1
4       CONTINUE
        IF(IFLG.NE.0) THEN
          DO 5 I=1,60
5         INNAM(I)=TMPNAM(I)
        ELSE
          DO 7 I=1,15
7         INNAM(I)=ACCNAM(I)
          DO 9 I=16,60
          IM=I-15
9         INNAM(I)=TMPNAM(IM)
        END IF
        TYPE 111,INNAM
 111    FORMAT(' ',60A)
        DO 10 I=60,1,-1
        IF(INNAM(I).NE.' ') GO TO 14
10      INNAM(I)=0
14      TYPE 106
106     FORMAT(1X,/,'$Enter number of frequency points (64, ',
        1   '128 or 256):  ')
        ACCEPT *,NFREQ
        IF(NFREQ.NE.64.AND.NFREQ.NE.128.AND.NFREQ.NE.256)
        1   GO TO 14
C
16      TYPE 107
107     FORMAT(1X,/,'$Enter number of phase points (64, ',
        1   '128 or 256):  ')
        ACCEPT *,NPHASE
        IF(NPHASE.NE.64.AND.NPHASE.NE.128.AND.NPHASE.NE.256)
        1   GO TO 16
C
18      TYPE 108
108     FORMAT(1X,/,'$Enter number of echos(1, 4, 8, 16 ):  ')
        ACCEPT *,NECHO
        IF(NECHO.LE.0.OR.NECHO.GT.16) GOTO 18
C
        IF(NPHASE.GT.NFREQ) GO TO 910
        NFLG=NFREQ
        IF(NFLG.EQ.64) THEN
          NBLOCK=16
          MATTYP='4'
        ELSE IF(NFLG.EQ.128) THEN
          NBLOCK=64
          MATTYP='6'
        ELSE
          NBLOCK=256
          MATTYP='8'
        END IF
C
C Enter study parameters
C
C
 205    CONTINUE
        CALL STUDAT  ! Input study parameters
C
C Read .RAW file (unit = 7) into virtual array one echo at a time.
C Calculate the mode of image desired and convert to INTEGER*2,
C then write to output file (unit = 2).
C Finally, close all files.
C
        CALL CALC(INNAM)
C
C
 200    CONTINUE
        TYPE *,'Further processing on this file ? (Y or N)'
        ACCEPT 201,ANS
 201    FORMAT(A)
        IF(ANS.NE.'Y'.AND.ANS.NE.'N') GOTO 200
        IF(ANS.EQ.'N') THEN
          TYPE 118
 118      FORMAT('0*** Reconstruction Completed! ***',/)
          GO TO 1
        ELSE
```

```
            GO TO 205
          END IF
C
C Error routines
C
900       TYPE 902
902       FORMAT('0** ERROR ** Unable to open input image file')
          CLOSE (UNIT=1)
          STOP
C
910       TYPE 912
912       FORMAT('0** ERROR ** The number of frequency points '
          1     'must be greater than ',/,13x,
          2     'or equal to the number of phase points!',/)
          CLOSE (UNIT=1)
          STOP
          END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
          SUBROUTINE OUTOPN( NE )
C
C This subroutine sets up the output file parameters (name,
C size, etc.), opens the file, and write the initial
C administrative blocks.
C
          LOGICAL*1 BUFFL(512)
          INTEGER*2 BUFF(256)
          LOGICAL*1 PNAM(16),PNUM(14),ORGAN(10),VIEW(7),STDATE(9)
          LOGICAL*1 OUTFNU(2),MATTYP,OUTNAM(31),COMMA,SPACE,EOS
          CHARACTER*32 ANSWER
          BYTE ANS(32)
          EQUIVALENCE ( ANSWER, ANS(1) )
          EQUIVALENCE (BUFF(1),BUFFL(1))
          COMMON /AA/ PNAM,PNUM,ORGAN,VIEW,STDATE,OUTFNU
          COMMON /BB/ BUFF
          COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
          DATA  OUTNAM/'[','S','T','E','V','E','N','.','I','M','A',
     *        'G','E','S',']',16*0/
          DATA COMMA,SPACE,EOS/',',' ',"200/
C
C Setup file name
C
          DO 10 I=16,31
          OUTNAM(I)=0
 10    CONTINUE
C
          J=0
          I=16
          DO 14 K=1,8
12        J=J+1
          IF(J.GT.16) GO TO 16
          IF((PNAM(J).GE.'A'.AND.PNAM(J).LE.'Z').OR.
     *      (PNAM(J).GE.'0'.AND.PNAM(J).LE.'9')) THEN
            OUTNAM(I)=PNAM(J)
            I=I+1
            GO TO 14
          END IF
          GO TO 12
14        CONTINUE
C
16        OUTNAM(I)='.'
          OUTNAM(I+1)='X'
```

```
        OUTNAM(I+2)=OUTFNU(1)
        OUTNAM(I+3)=OUTFNU(2)+NE
        OUTNAM(I+4)=0
C
C Open output file.
C
        OPEN (UNIT=2,FILE=OUTNAM,TYPE='NEW',ACCESS='DIRECT',
        1         RECL=128,INITIALSIZE=NBLOCK+6,ERR=910)
C
C Setup and write header block
C
        DO 20 I=1,256
20      BUFF(I)=0
C
        BUFF(1)=1
        BUFF(2)=NBLOCK+6
        BUFFL(7)=MATTYP-"60
        BUFF(6)=1
        BUFF(7)=1
        BUFF(9)=3
        BUFF(10)=4
        BUFF(19)=7
        BUFF(20)=1
C
        I=176
        DO 22 J=1,16
        K=J+I
22      BUFFL(K)=PNAM(J)
        BUFFL(193)=COMMA
        BUFFL(194)=SPACE
        I=194
        DO 24 J=1,14
        K=J+I
24      BUFFL(K)=PNUM(J)
        BUFFL(209)=COMMA
        BUFFL(210)=SPACE
        I=210
        DO 26 J=1,10
        K=J+I
26      BUFFL(K)=ORGAN(J)
        BUFFL(221)=COMMA
        BUFFL(222)=SPACE
        I=222
        DO 28 J=1,7
        K=J+I
28      BUFFL(K)=VIEW(J)
        BUFFL(230)=COMMA
        BUFFL(231)=SPACE
        BUFFL(232)='U'
        BUFFL(233)='S'
        DO 30 K=234,236
30      BUFFL(K)=SPACE
        BUFFL(237)='1'
        BUFFL(238)=':'
        BUFFL(239)=MATTYP
        BUFFL(240)=COMMA
        BUFFL(241)=SPACE
        I=241
        DO 32 J=1,9
        K=J+I
32      BUFFL(K)=STDATE(J)
        BUFFL(251)=COMMA
        BUFFL(252)=SPACE
        BUFFL(253)='.'
```

```
      BUFFL(254)='X'
      BUFFL(255)=OUTFNU(1)
      BUFFL(256)=OUTFNU(2)
C
      WRITE (2'1,ERR=965) BUFF
C
C Open dummy GAMMA-11 header file, modify, and write to
C administration block.
C
      TYPE *,' '
      TYPE *,' Echo number',NE
      TYPE *,' '
      IF(NFLG.NE.256) THEN
         OPEN (UNIT=3,FILE='[STEVEN.ASPECT]DUMDUM.XXX',TYPE='OLD',
     1         ACCESS='DIRECT',RECL=128,ERR=920)
      ELSE
         OPEN (UNIT=3,FILE='[STEVEN.ASPECT]DUMDUM.256',TYPE='OLD',
     1         ACCESS='DIRECT',RECL=128,ERR=920)
      END IF
C
      READ (3'1,ERR=970) BUFF
      CLOSE (UNIT=3)                    ! Close dummy header file
      BUFFL(243)=MATTYP
      BUFF(98)=NBLOCK
C
      I=1
      DO 40 J=1,16
      K=I+J
40    BUFFL(K)=PNAM(J)
      BUFFL(18)=EOS
      DO 42 K=19,23
42    BUFFL(K)=0
      BUFFL(24)=1
      I=24
      DO 44 J=1,14
      K=I+J
44    BUFFL(K)=PNUM(J)
      BUFFL(39)=EOS
      BUFFL(96)=1
      I=96
      DO 46 J=1,10
      K=I+J
46    BUFFL(K)=ORGAN(J)
      BUFFL(107)=EOS
      BUFFL(108)=0
      BUFFL(109)=0
      BUFFL(110)=1
      I=110
      DO 48 J=1,7
      K=I+J
48    BUFFL(K)=VIEW(J)
      BUFFL(118)=EOS
      BUFFL(119)=0
      BUFFL(53)=1
      I=53
      DO 50 J=1,8
      K=I+J
50    BUFFL(K)=STDATE(J)
      BUFFL(62)=EOS
C
      WRITE (2'2,ERR=965) BUFF
C
C Write four (4) blank comment blocks
C
```

```
        DO 52 I=1,256
52      BUFF(I)=0
        WRITE (2'3,ERR=965) BUFF
        WRITE (2'4,ERR=965) BUFF
        WRITE (2'5,ERR=965) BUFF
        WRITE (2'6,ERR=965) BUFF
C
        RETURN
C
C Error routines
C
910     TYPE 912
912     FORMAT('0** ERROR ** Unable to open output image file')
        GOTO 999
920     TYPE 922
922     FORMAT('0** ERROR ** Unable to open file DUMDUM.XXX')
        GOTO 999
C
965     TYPE 967
967     FORMAT('0** ERROR ** Error writing output image file')
        GOTO 999
970     TYPE 972
972     FORMAT('0** ERROR ** Error reading file DUMDUM.XXX')
        GOTO 999
 1000   FORMAT(A)
C
999     CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        CLOSE (UNIT=4)
        STOP
        END

C
C
        SUBROUTINE STUDAT
C
C This subroutine is used to enter study or patient information.
C
        IMPLICIT INTEGER*4 (I-N)
        CHARACTER*26 PRPNAM,PRPNUM,PRORGA,PRVIEW,PRSTUD,PROUTF
        LOGICAL*1 PNAM(16),PNUM(14),ORGAN(10),VIEW(7),STDATE(9)
        LOGICAL*1 OUTFNU(2),ANS
        COMMON /AA/ PNAM,PNUM,ORGAN,VIEW,STDATE,OUTFNU
        DATA PRPNAM/'Study or patient name:    '/
        DATA PRPNUM/'Study or patient number:  '/
        DATA PRORGA/'Organ:                    '/
        DATA PRVIEW/'View:                     '/
        DATA PRSTUD/'Study date:               '/
        DATA PROUTF/'Output file number (##):  '/
C
C Input initial data
C
        TYPE 100
100     FORMAT('0INPUT THE FOLLOWING STUDY DOCUMENTATION',/)
102     FORMAT('$',A26)
103     FORMAT(16A1)
        TYPE 102,PRPNAM
        ACCEPT 103,PNAM
        CALL LOWUPP(PNAM)
        TYPE 102,PRPNUM
        ACCEPT 103,PNUM
        CALL LOWUPP(PNUM)
        TYPE 102,PRORGA
        ACCEPT 103,ORGAN
        CALL LOWUPP(ORGAN)
        TYPE 102,PRVIEW
        ACCEPT 103,VIEW
        CALL LOWUPP(VIEW)
        TYPE 102,PRSTUD
        ACCEPT 103,STDATE
        CALL LOWUPP(STDATE)
        STDATE(9)=' '             ! Necessary for compatibility with
                                  ! GAMMA-11
        TYPE 102,PROUTF
        ACCEPT 103,OUTFNU
        CALL LOWUPP(OUTFNU)
        IF(OUTFNU(1).LT."60.OR.OUTFNU(1).GT."71) THEN
          OUTFNU(1)='0'
          OUTFNU(2)='0'
        END IF
        IF(OUTFNU(2).LT."60.OR.OUTFNU(2).GT."71) THEN
          OUTFNU(2)=OUTFNU(1)
          OUTFNU(1)='0'
        END IF
C
C Verify input and correct any errors
C
10      TYPE 104
104     FORMAT(1X,///,'0VERIFY THE FOLLOWING STUDY PARAMETERS',/)
105     FORMAT(1X,1A4,1A26,16A1)
        TYPE 105,'(1) ',PRPNAM,PNAM
        TYPE 105,'(2) ',PRPNUM,PNUM
        TYPE 105,'(3) ',PRORGA,ORGAN
        TYPE 105,'(4) ',PRVIEW,VIEW
        TYPE 105,'(5) ',PRSTUD,STDATE
        TYPE 105,'(6) ',PROUTF,OUTFNU
12      TYPE 106
106     FORMAT(1X,/,'$      Any errors (Y or N)?  ')
        ACCEPT 103,ANS
        CALL LOWUPP(ANS)
        IF(ANS.NE.'Y'.AND.ANS.NE.'N') GO TO 12
        IF(ANS.EQ.'N') THEN
          TYPE 107
107       FORMAT(2X)
          RETURN
        END IF
C
        TYPE 108
108     FORMAT(1X,/,'$Which line do you wish to correct?  ')
        ACCEPT 110,LINE
110     FORMAT(I8)
        IF(LINE.EQ.1) THEN
          TYPE 102,PRPNAM
          ACCEPT 103,PNAM
          CALL LOWUPP(PNAM)
        ELSE IF(LINE.EQ.2) THEN
          TYPE 102,PRPNUM
          ACCEPT 103,PNUM
          CALL LOWUPP(PNUM)
        ELSE IF(LINE.EQ.3) THEN
          TYPE 102,PRORGA
          ACCEPT 103,ORGAN
          CALL LOWUPP(ORGAN)
        ELSE IF(LINE.EQ.4) THEN
          TYPE 102,PRVIEW
          ACCEPT 103,VIEW
          CALL LOWUPP(VIEW)
        ELSE IF(LINE.EQ.5) THEN
          TYPE 102,PRSTUD
          ACCEPT 103,STDATE
          CALL LOWUPP(STDATE)
          STDATE(9)=' '   ! Necessary for compatibility with
                                      ! GAMMA-11
        ELSE IF(LINE.EQ.6) THEN
          TYPE 102,PROUTF
          ACCEPT 103,OUTFNU
          CALL LOWUPP(OUTFNU)
          IF(OUTFNU(1).LT."60.OR.OUTFNU(1).GT."71) THEN
                OUTFNU(1)='0'
                OUTFNU(2)='0'
          END IF
          IF(OUTFNU(2).LT."60.OR.OUTFNU(2).GT."71) THEN
                OUTFNU(2)=OUTFNU(1)
                OUTFNU(1)='0'
          END IF
        END IF
        GO TO 10
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
        SUBROUTINE LOWUPP(DUM)
C
C This subroutine converts a string of lower case characters to upper
C case.
C
        IMPLICIT INTEGER*4 (I-N)
        LOGICAL*1 DUM(16)
C
        DO 10 I=1,16
        IF(DUM(I).GE."141.AND.DUM(I).LE."172) DUM(I)=DUM(I)-"40
10      CONTINUE
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCCC
C
        SUBROUTINE CALC(ISTRING)
C
C Read raw processed data file (unit = 7) into virtual array.
C Convert to INTEGER*2, process by real, imaginary , phase ,
C or modulus then write to output file (unit = 2).
C Finally, close all files.
C
        IMPLICIT INTEGER*4 (I-N)
        LOGICAL*1 MATTYP,ISTRING(60)
        INTEGER*2 BUFFW(256),IWDUM, TEMPBUF(256,256)
        INTEGER*4 BUFFI(128),APBASE,APADDR,ATAB1,ATAB2,ATAB3,ATAB4,
1               NECHO,BUFF(128)
        REAL*4 BUFFR(128)
        COMPLEX*8 BUFFC(64),ZMAT(256,256),CDUM,Z2MAT(256,256)
        COMPLEX ZTEMPC(256)
        REAL ZTEMPR(512)
        CHARACTER*32 ANSWER
        BYTE ANS(32)
        BYTE NAME(60)
        CHARACTER*60 NSTRING
        EQUIVALENCE (NSTRING,NAME(1))
        EQUIVALENCE ( ANSWER, ANS(1) )
```

```
         CHARACTER*1 SAMPLE, MODE, ALREADY
C
C
         EQUIVALENCE (ZTEMPC(1),ZTEMPR(1))
         EQUIVALENCE (BUFFI(1),BUFFW(1),BUFFR(1),BUFFC(1))
C
C
         COMMON /BB/ BUFF,N_BLOCK,M_BLOCK,SAMPLE,NECHO,ALREADY
         COMMON /CC/ NFLG,NBLOCK,NFREQ,NPHASE,MATTYP,ANSWER
C
C copy parameter passed
C
         DO I = 1, 60
           NAME(I) = ISTRING(I)
         END DO
C
C get proccessing mode
C
 100     TYPE *,'Enter the type of image you want to see;'
         TYPE *,' '
         TYPE *,'            R for R(EAL)'
         TYPE *,'            I for I(MAGINARY)'
         TYPE *,'            P for P(HASE)'
         TYPE *,'            M for M(ODULUS)'
         TYPE *,' '
         READ(5,1000) ANSWER
         IF(ANS(1).NE.'R'.AND.ANS(1).NE.'I'.AND.
         1  ANS(1).NE.'P'.AND.ANS(1).NE.'M') THEN
           TYPE *,'***** Error in input. Try again...'
           GOTO 100
         END IF
         IF(ANS(1).EQ.'R') ANSWER = 'REAL'
         IF(ANS(1).EQ.'I') ANSWER = 'IMAGINARY'
         IF(ANS(1).EQ.'P') ANSWER = 'PHASE'
         IF(ANS(1).EQ.'M') ANSWER = 'MODULUS'
C
C Open a file to contain the complex arrays; 1 array per echo
C
         DO I = 1,60
           IF(NAME(I).EQ.'.') IPERIOD = I
         END DO
         NAME(IPERIOD+1) = 'R'
         NAME(IPERIOD+2) = 'A'
         NAME(IPERIOD+3) = 'W'
         TYPE *,' IN CALC. NAME = ',NSTRING
         OPEN( UNIT=7,FILE=NAME,FORM='UNFORMATTED',STATUS='OLD',ERR=900 )
C
C process for NECHO echos
C
         DO NE = 1,NECHO
C
C Zero virtual array
C
          DO I = 1, 512
          ZTEMPR(I) = 0.0
          END DO
C
          DO 12 J=1,NFLG
          DO 10 I=1,NFLG
10        ZMAT(I,J)=(0.0,0.0)
12        CONTINUE
C
C get the current echo
C
```

```
       READ(7) ZMAT
C
       NFR=NFREQ/64
       NPHHA=NPHASE/2
       NFRHA=NFREQ/2
       NFLHA=NFLG/2
C
       XMAX = -1.E28
       XMIN =  1.E28
C
C process the image as indicated in ANSWER
C
      IF(ANSWER.EQ.'REAL                ') THEN
       DO J=1,NFLG
        DO I=1,NFLG
         TEMP = REAL( ZMAT(I,J) )
         IF(TEMP.GT.XMAX) XMAX=TEMP
            IF(TEMP.LT.XMIN) XMIN=TEMP
         ZMAT(I,J) = CMPLX( TEMP, 0.0 )
        END DO
       END DO
      ELSEIF(ANSWER.EQ.'IMAGINARY           ') THEN
       DO J=1,NFLG
        DO I=1,NFLG
            TEMP = AIMAG( ZMAT(I,J) )
         IF(TEMP.GT.XMAX) XMAX=TEMP
            IF(TEMP.LT.XMIN) XMIN=TEMP
         ZMAT(I,J) = CMPLX( TEMP, 0.0 )
        END DO
       END DO
      ELSEIF(ANSWER.EQ.'MODULUS             ') THEN
       DO J=1,NFLG
        DO I=1,NFLG
            TEMP = CABS( ZMAT(I,J) )                   ! find the modulus
         IF(TEMP.GT.XMAX) XMAX=TEMP
            IF(TEMP.LT.XMIN) XMIN=TEMP
         ZMAT(I,J) = CMPLX( TEMP, 0.0 )
        END DO
       END DO
      ELSEIF(ANSWER.EQ.'PHASE               ') THEN
       PI = 3.1415926
       DO J=1,NFLG
        DO I=1,NFLG
            IF(CABS(ZMAT(I,J)).LT.1000.0) THEN
              ZMAT(I,J) = CMPLX( 0.0, 0.0 )
            ELSE
          TEMP1 = REAL( ZMAT(I,J) )
              IF(TEMP1.LE.1.E-20.AND.TEMP1.GE.-1.E-20) TEMP1=1.0E-20
              TEMP2 = AIMAG( ZMAT(I,J) )
              TEMP = ATAN( TEMP2/TEMP1 )               ! find the phase angle
              IF(TEMP1.LT.0.0.AND.TEMP2.LT.0.0) TEMP=TEMP-PI
              IF(TEMP1.LT.0.0.AND.TEMP2.GT.0.0) TEMP=TEMP+PI
          ZMAT(I,J) = CMPLX( TEMP, 0.0 )
            END IF
        END DO
       END DO
      END IF
C
C
      IF(ANSWER.EQ.'PHASE               ') THEN
       XMAX = 3.1415926
       XMIN = -1.*XMAX
      END IF
      TYPE *,'XMIN=',XMIN,' XMAX=',XMAX
```

```
        CON32=32767.
        XNORM=CON32/(XMAX-XMIN)
        TYPE 111,XNORM
111     FORMAT('0Normalization factor = 'E12.5)
        DO J=1,NFLG
          DO I=1,NFLG
            TEMPBUF(I,J)=INT((REAL(ZMAT(I,J))-XMIN)*XNORM)
          END DO
        END DO
C
C
C Write data to output buffer, converting to INTEGER*2
C in the process, then write to output file.
C
        CALL OUTOPN( NE )
        IREC=6               ! Disk block counter
        DO J=1,NFLG
          DO I=NFLG,1,-1
            ICNT=ICNT+1
            BUFFW(ICNT)=TEMPBUF(I,J)
            IF(ICNT.EQ.256) THEN
              IREC=IREC+1
              WRITE (2'IREC,ERR=965) BUFFW
              ICNT=0
            END IF
          END DO
        END DO
        CLOSE(2)
        END DO
C
        CLOSE(7)
        RETURN
C
1000    FORMAT(A)
C
C
C Error routines
C
965     TYPE 967
967     FORMAT('0** ERROR ** Error writing output image file')
        GOTO 999
900     TYPE 982
982     FORMAT('0** ERROR ** Error reading file DUMDUM.TP1')
        GOTO 999
C
999     CLOSE (UNIT=1)
        CLOSE (UNIT=2)
        CLOSE (UNIT=3)
        CLOSE (UNIT=4)
        STOP
        END
```

Appendix N

The region of interest analysis program PHASME.

```
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCC
C
      SUBROUTINE GRINEND
C
C Subroutine to release the GRINNELL
C
   INTEGER*2 GR_BUFF(1024),GR_LEN,CHNMSK,OVLMSK,MON
   COMMON /GRBUFR/GR_LEN,GR_BUFF
   DATA GR_LEN/1024/
   INTEGER*4 IERR
C
C Call GRSEND to release I/O channel to GRINNELL
C
   CALL GRSEND
C
C Call GRELEASE to release GRINNELL display channel
C
   CALL GRELEASE(MON)
      RETURN
      END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC
C
  SUBROUTINE PREP_CURSOR
C
C     Set up the cursor unit to the grinnell. i.e. clear the
C     cursor unit
C
  INTEGER*2 X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4)
C
  COMMON /FLAGS/X_CURS,Y_CURS,ENTER,F1,F2
C
C read the cursor back once to clear the ENTER flag
C
      ENTER(1) = 0
C
C Make sure cursor 1 is turned on.
C
    DO WHILE (ENTER(1).EQ.0) !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
       WRITE(6,500)                                                !
       CALL GRQWT(1,STATUS)                                        !
       CALL GRQRD(1,1,X_CURS(1),Y_CURS(1),ENTER(1),F1(1),F2(1))    !
       CALL GRQRD(1,2,X_CURS(2),Y_CURS(2),ENTER(2),F1(2),F2(2))    !
       CALL GRQRD(1,3,X_CURS(3),Y_CURS(3),ENTER(3),F1(3),F2(3))    !
       CALL GRQRD(1,4,X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4))    !
     END DO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
C Make sure both function flags are turned off
    DO WHILE (F1(1).NE.0.OR.F2(1).NE.0) !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
       WRITE(6,501)                                                !
```

```
         CALL GRQWT(1,STATUS)                                         !
         CALL GRQRD(1,1,X_CURS(1),Y_CURS(1),ENTER(1),F1(1),F2(1))     !
         CALL GRQRD(1,2,X_CURS(2),Y_CURS(2),ENTER(2),F1(2),F2(2))     !
         CALL GRQRD(1,3,X_CURS(3),Y_CURS(3),ENTER(3),F1(3),F2(3))     !
         CALL GRQRD(1,4,X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4))     !
       END DO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
   RETURN
 500 FORMAT(/,' TURN ON CURSOR 1 AND HIT ENTER FLAG',/)
 501 FORMAT(/,' TURN OFF BOTH FUNCTION FLAGS AND HIT ENTER FLAG',/)
   END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCC
C
   SUBROUTINE CURSOR_MARK
C
C  A routine to define an area on the screen enclosed by a rectangle.
C The user defines one corner of the box and then,  using the track ball
C unit, moves the cursor to a position on the screen where the opposite
C corner of the box is to be.  At each movement of the cursor the previously
C drawn box is erased and a new box,  as defined by the new cursor position,
C is drawn in its stead.
C
        INTEGER*2 X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4)
        CHARACTER*1 ANSWER
C
        INTEGER*2 IMGBUF(65536), GR_BUFF(1024), GR_LEN, CHNMSK, OVLMSK
        1 MON, SUBMSK, LINE(20), BCKGND, REPMOD,  N(16),REALBF(65536)
        !INTEGER*4 IERR, X, Y, ROW, COL, LOWER, UPPER, X_INC, Y_INC,
        1 LENGTH, DBL_W, DBL_H, ZERO_C, XSTART, YSTART, TOP, BOTTOM,
        2 BOXES(20,4),YEND,XEND
C
        LOGICAL SWITCH
C
        COMMON/GRBUFR/ GR_LEN, GR_BUFF
        COMMON/GRVARI/ CHNMSK, OVLMSK, SUBMSK, MON
        COMMON/ARRAY/ REALBF,IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
        COMMON/BOX/ BOXES
        COMMON /FLAGS/X_CURS,Y_CURS,ENTER,F1,F2
        DATA GR_LEN/1024/, BCKGND/0/, ZERO_C/1/, SUBMSK/'600'X/, OVLMSK/'600'X/
C
C Clear array that will contain the box coordinates
C
        DO J = 1,4
          DO I = 1,20
            BOXES(I,J) = 0
          END DO
        END DO
C
        NBOX_NUMBER = 1
C
C Gets coordinates of cursor from grinnell.
C
 300    CONTINUE
C
C User moves cursor 1 to desired position, turns on either function flag
C and hits enter flag.
C
        F1(1) = 0
        F2(1) = 0
C
   WRITE(6,502)
   CALL GRFER(OVLMSK,OVLMSK,0)        !Erase overlay channel
```

```
C
   DO WHILE (F1(1).EQ.0.AND.F2(1).EQ.0) !!!!!!!!!!!!!!!!!!!!!!!!!!!!
       CALL GRQWT(1,STATUS)                                              !
       CALL GRQRD(1,1,X_CURS(1),Y_CURS(1),ENTER(1),F1(1),F2(1))          !
       CALL GRQRD(1,2,X_CURS(2),Y_CURS(2),ENTER(2),F1(2),F2(2))          !
       CALL GRQRD(1,3,X_CURS(3),Y_CURS(3),ENTER(3),F1(3),F2(3))          !
       CALL GRQRD(1,4,X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4))          !
    END DO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
      NOLDX = X_CURS(1)
      NOLDY = Y_CURS(1)
      NXPOINT = X_CURS(1)
      NYPOINT = Y_CURS(1)
C
C  now monitor the cursor position and draw a box interactively
C
      WRITE(6,503)
      WRITE(6,504)
      F1(1) = 0
      F2(1) = 0
      DO WHILE  (F1(1).EQ.0.AND.F2(1).EQ.0) !!!!!!!!!!!!!!!!!!!!!!!!!!!!
        CALL GRQWT(1,STATUS)                                                !
        CALL GRQRD(1,1,X_CURS(1),Y_CURS(1),ENTER(1),F1(1),F2(1))       !
        IF(X_CURS(1).NE.NOLDX.OR.Y_CURS(1).NE.NOLDY) THEN                   !
          CALL GRFER(OVLMSK,OVLMSK,0)        !Erase overlay channel  !
        END IF                                                     !
        NOLDX = X_CURS(1)                                              !
        NOLDY = Y_CURS(1)                                              !
        XSTART = NXPOINT                     !Draw the box             !
        YSTART = NYPOINT                                               !
        XEND = X_CURS(1)                                               !
        YEND = YSTART                                                  !
        CALL GRFVC(OVLMSK,SUBMSK,BCKGND,ZERO_C,XSTART,YSTART,XEND,YEND)!
        XSTART = XEND                                                  !
        YEND = Y_CURS(1)                                               !
        CALL GRFVC(OVLMSK,SUBMSK,BCKGND,ZERO_C,XSTART,YSTART,XEND,YEND)!
        XEND = NXPOINT                                                 !
        YSTART = YEND                                                  !
        CALL GRFVC(OVLMSK,SUBMSK,BCKGND,ZERO_C,XSTART,YSTART,XEND,YEND)!
        XSTART = XEND                                                  !
        YEND = NYPOINT                                                 !
        CALL GRFVC(OVLMSK,SUBMSK,BCKGND,ZERO_C,XSTART,YSTART,XEND,YEND)!
        CALL GRSBFD                                                    !
        ZAREA = ABS((NYPOINT-Y_CURS(1))/2)*ABS((NXPOINT-X_CURS(1))/2)  !
        TYPE *,'Current area = ',ZAREA                             !
      END  DO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
C
C now we have the four points defining the box, so save them in BOXES
C
      BOXES(NBOX_NUMBER,1) = NYPOINT
      BOXES(NBOX_NUMBER,2) = NXPOINT
      BOXES(NBOX_NUMBER,3) = Y_CURS(1)
      BOXES(NBOX_NUMBER,4) = X_CURS(1)
C
      IF(NBOX_NUMBER.GE.20) RETURN
      TYPE *,' '
      TYPE *,' Area of this box is = ',ZAREA
      TYPE *,' '
      TYPE *,' Happy with this box ?'
      READ(5,1000) ANSWER
      IF(ANSWER.NE.'Y') GOTO 300
      TYPE *,' Another box ( Y or N ) ?'
400   CONTINUE
      READ(5,1000) ANSWER
      IF(ANSWER.NE.'Y'.AND.ANSWER.NE.'N') GOTO 400
```

```
          IF(ANSWER.EQ.'Y') THEN
            TYPE *,' Same procedure as before...'
            NBOX_NUMBER = NBOX_NUMBER + 1
            GOTO 300
          END IF
   RETURN
C
 502 FORMAT(/,' Position cursor 1 at the top left point of the area'
          1     ,/,' of interest, Turn on either function flag, and hit the'
          2     ,/,' ENTER flag')
503       FORMAT(/,' Turn off the function flags')
504       FORMAT(/,' Now use the track ball WITH TRACK SWITCH ON to define'
          1     ,/,' the box over the area of interest. When happy turn on a'
          2     ,/,' function flag and hit enter.')
C
 1000     FORMAT(A)
   END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
          SUBROUTINE DRAW_BOX
C
C  A subroutine to draw a box about a selected area on the GRINNELL.
C
C  Up to 20 such areas may be defined. These areas are stored in an array
C  as follows ( assuming the user starts with the top left corner and
C  proceed to the bottom right; otherwise this is just academic );
C
C         INTEGER*4 BOXES( 20, 4 )
C
C         BOXES(1,1) = top most vertical point
C     BOXES(1,2) = left most horizontal point
C     BOXES(1,3) = bottom vertical point
C         BOXES(1,4) = right horizontal point
C
C
          INTEGER*2 IMGBUF(65536), GR_BUFF(1024), GR_LEN, CHNMSK, OVLMSK
          1 MON, SUBMSK, LINE(20), BCKGND, REPMOD, N(16), REALBF(65536)
          INTEGER*4 IERR, X, Y, ROW, COL, LOWER, UPPER, X_INC, Y_INC,
          1 LENGTH, DBL_W, DBL_H, ZERO_C, XSTART, YSTART, TOP, BOTTOM,
          2 BOXES(20,4)
C
          LOGICAL SWITCH
C
          COMMON/GRBUFR/ GR_LEN, GR_BUFF
          COMMON/GRVARI/ CHNMSK, OVLMSK, SUBMSK, MON
          COMMON/ARRAY/ REALBF,IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
          COMMON/BOX/ BOXES
          DATA GR_LEN/1024/
C
          CALL PREP_CURSOR
          CALL CURSOR_MARK
          RETURN
          END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
          SUBROUTINE DRAW( X, Y, MODE )
C
C  A subroutine to place the image on the GRINNELL
C
          INTEGER*2 IMGBUF(65536), GR_BUFF(1024), GR_LEN, CHNMSK, OVLMSK
```

```
        1 MON, SUBMSK, LINE(20), BCKGND, REPMOD,  N(16), REALBF(65536)
        INTEGER*4 IERR, X, Y, ROW, COL, LOWER, UPPER, X_INC, Y_INC,
        1 LENGTH, DBL_W, DBL_H, ZERO_C, XSTART, YSTART, TOP, BOTTOM,
        2 MODE
C
        LOGICAL SWITCH
C
        COMMON/GRBUFR/ GR_LEN, GR_BUFF
        COMMON/GRVARI/ CHNMSK, OVLMSK, SUBMSK, MON
        COMMON/ARRAY/ REALBF, IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
        DATA GR_LEN/1024/
C
C Check mode switch. If equal to 1, then skip the allocation and setup
C of the Grinnell
C
        IF(MODE.EQ.2) GOTO 50
C
C Allocate a GRINNELL display channel (monitor) to this process. MON
C is set to 0 to request an interactive channel allocation
C
    MON=0
    CALL GRSHARE(CHNMSK,OVLMSK,MON)
    IF (MON.LT.0) STOP
C
C If the color channel was assigned, change the default overlay channel
C mask from '200'X (blue overlay channel only) to 'E00'X (blue,green and
C white overlay channels) so that all overlay channels may be erased
    IF (OVLMSK.EQ.'200'X) OVLMSK='E00'X
C
C Call GRSINI to assign a channel to the GRINNELL
    CALL GRSINI
C
    CALL GRFER(CHNMSK,'FF'X,0)  !Erase display channel
    CALL GRFER(OVLMSK,OVLMSK,0)         !Erase overlay channel
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
 50      CONTINUE
C
C Set constants for GRINNELL calls
C
        LENGTH = ROW
        ZERO_C = 1
        X_INC = 1
        Y_INC = 0
        DBL_H = 0
        DBL_W = 0
        NYSTEP = 1
        IF(ROW.LE.256.AND.COL.LE.256) THEN
          DBL_H = 1
          DBL_W = 1
          NYSTEP = 2
          X_INC = 2
        END IF
        XSTART = X
        YSTART = Y
C
C  Now cycle through the image buffer using the row and col definitions
C  to set the size of the image produced
C
        DO 100 J = 1, COL
        K = ( J - 1 )*ROW + 1
        CALL GRWDW(IMGBUF(K),XSTART,YSTART,X_INC,Y_INC,LENGTH,
        1 DBL_W,DBL_H,CHNMSK,ZERO_C)
        YSTART = YSTART + NYSTEP
 100    CONTINUE
C
        CALL GRSBFD
C
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
        SUBROUTINE REPLACE_PIX( II, JJ )
C
C Routine to replace a pixel with a white ( 255 ) pixel to mark it as
C counted.
C
        INTEGER*2 IMGBUF(65536), GR_BUFF(1024), GR_LEN, CHNMSK, OVLMSK
        1 MON, SUBMSK, LINE(20), BCKGND, REPMOD,  N(16), DATA, REALBF(65536)
        INTEGER*4 IERR, X, Y, ROW, COL, LOWER, UPPER, X_INC, Y_INC,
        1 LENGTH, DBL_W, DBL_H, ZERO_C, XSTART, YSTART, II, JJ
C
        LOGICAL SWITCH
C
        COMMON/GRBUFR/ GR_LEN, GR_BUFF
        COMMON/GRVARI/ CHNMSK, OVLMSK, SUBMSK, MON
        COMMON/ARRAY/ REALBF,IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
        DATA GR_LEN/1024/
C
C get coordinates of point to white out
C
        XSTART = II
        YSTART = JJ
C
C constants for GRINNELL
C
        LENGTH = 1
        X_INC = 1
        Y_INC = 0
        DBL_W=0
        DBL_H=0
        ZERO_C=1
        DATA = 255
C
C white it on the GRINNELL
C
        CALL GRWDW( DATA,XSTART,YSTART,X_INC,Y_INC,LENGTH,DBL_H,
        1       DBL_W,CHNMSK,ZERO_C )
C
C BYE
C
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
        SUBROUTINE SCALE
C
C  This subroutine finds the range ( min , max ) of points
C  in an image data file of delta format and scales the entire
C  image array (IMGBUF).
C
        INTEGER*4 MAX, MIN, ROW, COL, K, J, I, LOWER, UPPER
        INTEGER*2 IMGBUF(65536), REALBF(65536)
        REAL SCALE_FAC
        LOGICAL SWITCH
C
        COMMON/ARRAY/ REALBF,IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
C
        MAX = -99999
        MIN = 99999
        DO 10 J=1,COL
        DO 10 I=1,ROW
C
C calculate image pixel position in the buffer
C
        K = ( J - 1 )*ROW + I
        IF(IMGBUF(K).GT.MAX) MAX=IMGBUF(K)
        IF(IMGBUF(K).LT.MIN) MIN=IMGBUF(K)
 10     CONTINUE
C
C now scale the entire buffer to values between 0 and 255
C
        SCALE_FAC = 255.0 / ( MAX - MIN )
        K = ROW * COL
        DO 20 I=1,K
        IMGBUF(I)=IMGBUF(I) * SCALE_FAC
 20     CONTINUE
        RETURN
        END
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C*********************************************************************
C
C       Program PHASME
C
C       Author: Steven Knudsen, NMR Group
C                                  U of A Hospital
C
C       last edit: 15/05/86
C
C  This program may be used to display images in DELTA format ( as from
C the Cross Center ) on the Dept. of Applied Science in Medicine's
C GRINNELL imaging system.
C
C I use the Grinnell track ball to mark the area of interest for a phase
C measurement. The user must input the size of that area and the number
C of images in the sequence. These images are assumed to have the suffix
C X01 through to Xxx where xx are the number of images in the sequence.
C ( e.g. MONK.X01, MONK.X02, MONK.X03, MONK.X04, etc. )
C
C The areas are assumed to be rectangular, therefore the user inputs the
C height and width of the area. This is done using the track ball unit to
C define a rectangle.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C  DELTA format images found in the following sizes;
C
C  Block ( 512 bytes )          Description of contents
C  ==================           =======================
C         1                     header block - contains
C                               image descriptors and
C                               constants
C         2                     administrative block
```

```
C       3                   expansion storage ( opt )
C          4                        comment block
C          6                        image data - sizes ;
C          .                                 32 X 32  byte
C          .                                 32 X 32  word
C          .                                 64 X 64  byte
C                                            64 X 64  word
C                                            128 X 128 byte
C                                            128 X 128 word
C                                            256 X 256 byte
C                                            256 X 256 word
C
C  Note that only the word formats are currently translated by
C this program.
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCC
C
C  Set up variables and constants
C
        INTEGER*2 IMGBUF(65536), BUF256(256), REALBF(65536)
C
     INTEGER*2 X_CURS(4),Y_CURS(4),ENTER(4),F1(4),F2(4)
        INTEGER*4 IERR, BLKREAD, I, J, IB, B, ROW, COL, LOWER, UPPER
        LOGICAL SWITCH, POINT, EVEN_ODD
        COMMON/ARRAY/ REALBF,IMGBUF, SWITCH, ROW, COL, LOWER, UPPER
     COMMON /FLAGS/X_CURS,Y_CURS,ENTER,F1,F2
     DATA GR_LEN/1024/
C
        INTEGER*2 GR_BUFF(1024), GR_LEN, CHNMSK, OVLMSK
        1 MON, SUBMSK
C
C Constants in the first block of the header
C
        INTEGER*2 FILTYP, TOTBLK, PROCFLG, IMGTYP, IISPAC, ISOFLG,
        1 MAXIMG, EXPSIZ,  COMOFF, COMSIZ, LOCBLK, LPBFLG, COLTBK,
        2 COLFLG, TEXTBK, TXTFLG, OVRLBK, OVLFLG, IMAGBK, IMGFLG,
        3 OVLMAP(4), IMGMAP(64), IMAGEBLK(256)
        INTEGER*4 DUMMY(128), BOXES(20,4), ECHO_INTENS(20,64), NUM_FILES
        CHARACTER*64 FILENAME, PRINTFILE
        BYTE FILE(64), PFILE(64)
        CHARACTER*1 ANSWER, HARD_SOFT
C
        LOGICAL END
C
        EQUIVALENCE( FILENAME, FILE(1) )
        EQUIVALENCE( PRINTFILE, PFILE(1) )
        COMMON/GRBUFR/ GR_LEN, GR_BUFF
        COMMON/GRVARI/ CHNMSK, OVLMSK, SUBMSK, MON
        COMMON/BOX/ BOXES
        COMMON/ECHO/ ECHO_INTENS, NUM_FILES
        DATA GR_LEN/1024/
C
C Initialize the BOXES and the ECHO_INTENS arrays
C
        DO J = 1,4
          DO I = 1, 20
            BOXES(I,J) = 0
          END DO
        END DO
        DO J = 1,64
          DO I = 1,20
            ECHO_INTENS(I,J) = 0
          END DO
        END DO
C
C Get image filename and open file for unformatted reads
C
        DO I = 1,64
          FILE(I) = ' '
          PFILE(I) = ' '
        END DO
        TYPE *, 'Enter the filename of the first image in the series'
        READ(5,600) FILENAME
C
C find the suffix of the file
C
        END = .FALSE.
        ICOLON = 0
        IPERIOD = -1
        DO I=64,1,-1
          IF(FILE(I).NE.' '.AND.(.NOT.END)) THEN
            END = .TRUE.
            IEND = I
          END IF
          IF(FILE(I).EQ.'.') THEN
            IPERIOD = I
          END IF
          IF(FILE(I).EQ.':') THEN
            ICOLON = I
          END IF
        END DO
        IF(IPERIOD.LE.1) GOTO 1001
        DO J = 1,IEND
          K = ICOLON + J
          PFILE(J) = FILE(K)
        END DO
        PFILE(IPERIOD+1-ICOLON) = 'P'
        PFILE(IPERIOD+2-ICOLON) = 'R'
        PFILE(IPERIOD+3-ICOLON) = 'T'
C
C Open a print out file for the data
C
        TYPE *,' '
        TYPE *,' Printout file is: ',PRINTFILE
        TYPE *,' '
        OPEN(UNIT=7,FILE=PRINTFILE,STATUS='NEW',FORM='FORMATTED')
C
C get the number of files in the series
C
 10     CONTINUE
        TYPE *,'Enter the number of files in the series'
        ACCEPT *,NUM_FILES
        IF(NUM_FILES.GT.100.OR.NUM_FILES.LT.1) THEN
          TYPE *,'**** ERROR. NUM_FILES=',NUM_FILES,' too large or small'
          GOTO 10
        END IF
C
C check if images made with hard 180 or soft 180. If soft the phase encoding
C will be the same for each echo... therefore no need to flip the even echos
C about the frequency axis
C
        HARD_SOFT = '?'
        DO WHILE(HARD_SOFT.NE.'H'.AND.HARD_SOFT.NE.'S')
          TYPE *,' H(ard) or S(oft) 180 pulses'
          READ(5,600) HARD_SOFT
        END DO
C
```

```
C now loop for the number of files in the series
C
         IMODE = 1
         POINT = .TRUE.
         EVEN_ODD = .TRUE.              ! Odd numbered echo in series
         DO IFILES = 1, NUM_FILES
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
C open the current file
C
         OPEN( UNIT=10, FILE=FILENAME, STATUS='OLD', FORM='UNFORMATTED',
         1   ERR=1001)
         TYPE *,' '
         TYPE *,'The current file is:',FILENAME
         TYPE *,' '
C
C and in the output file
C
         WRITE(7,700) FILENAME
 700     FORMAT(' ',/,' The file is: ',32A)
C
C read in image constants and descriptors
C
         READ(10) FILTYP, TOTBLK, PROCFLG, IMGTYP, IISPAC, ISOFLG,
         1 MAXIMG, EXPSIZ, COMOFF, COMSIZ, LOCBLK, LPBFLG, COLTBK,
         2 COLFLG, TEXTBK, TXTFLG, OVRLBK, OVLFLG, IMAGBK, IMGFLG,
         3 OVLMAP, IMGMAP, IMGMAP, IMGMAP, OVLMAP, OVLMAP, OVLMAP,
         4 OVLMAP, OVLMAP, OVLMAP, OVLMAP, OVLMAP, OVLMAP, OVLMAP
C
C        First block has been read so read five remaining blocks
C
         READ(10) DUMMY
         READ(10) DUMMY
         READ(10) DUMMY
         READ(10) DUMMY
         READ(10) DUMMY
C
C Determine which image size we are working with
C
         IF(IMGTYP.EQ.2) THEN
        BLKREAD = 4
           ROW = 32
           COL = 32
       ELSE IF(IMGTYP.EQ.4) THEN
           BLKREAD  = 16
           ROW = 64
           COL = 64
          ELSE IF(IMGTYP.EQ.6) THEN
           BLKREAD = 64
           ROW = 128
           COL = 128
          ELSE IF(IMGTYP.EQ.8) THEN
           BLKREAD = 256
           ROW = 256
           COL = 256
          ELSE
           STOP
         END IF
         TYPE *,' '
         TYPE *,'Image is ',ROW,' X ',COL
         TYPE *,' '
C
C and in the output file
```

```
C
          IF(POINT) THEN
            WRITE(7,701) ROW,COL
 701        FORMAT('  Image is ',I4,' X ',I4)
          END IF
C
C read in the image data from the file
C
          DO 1000 B=1,BLKREAD
          READ(10) BUF256
          DO 900 I=1,256
          IB = ( B - 1 )*256 + I
          IMGBUF(IB) = BUF256(I)
 900      CONTINUE
 1000     CONTINUE
          DO ICOPY = 1,65536
            REALBF(ICOPY) = IMGBUF(ICOPY)
          END DO
C
C close the file
C
          CLOSE(10)
C
C now check to see if the image should be reversed about the frequency axis
C
          IF(.NOT.EVEN_ODD.AND.HARD_SOFT.EQ.'H') THEN
            DO JY = 1,COL
              DO IX = 1, ROW/2-1
                IFLIPX = ROW - IX
                K = ( JY - 1 )*ROW + IX
                KFLIP = ( JY - 1 )*ROW + IFLIPX
                IDUM = IMGBUF(K)
                IMGBUF(K) = IMGBUF( KFLIP )
                IMGBUF( KFLIP ) = IDUM
              END DO
            END DO
          END IF
C
C  now scale the image to pixel values between 0 and 255
C
          CALL SCALE
C
C  display the image on the GRINNELL ( MODE = 1 means setup GRINNELL )
C
          CALL DRAW( 0, 0, IMODE )
C
C check for input of coords
C
          IF(POINT) THEN
C
C get regions of interest
C
            CALL DRAW_BOX
            WRITE(7,501)
 501        FORMAT(/,' BOX # | AREA      | AVERAGE INTENSITY | CALC PHASE')
            WRITE(7,502)
 502        FORMAT('========================================')
          END IF
C
C calculate the average pixel value in the area of interest
C
          IBOX = 1
          NDATA = BOXES( IBOX, 1 )
          DO WHILE(NDATA.NE.0)
          XSTART = BOXES(IBOX,2)/2
          XEND = BOXES(IBOX,4)/2
          YSTART = BOXES(IBOX,1)/2
          YEND = BOXES(IBOX,3)/2
          NXSTEP = 1
          NYSTEP = 1
          IF(XSTART.GT.XEND) NXSTEP = -1
          IF(YSTART.GT.YEND) NYSTEP = -1
          AREA = 0.0
          SUM = 0.0
C
C see if the image is reversed. If so then read the info from the box
C from the 'other half' of the image, i.e. reverse the read procedure
C
          IF(.NOT.EVEN_ODD.AND.HARD_SOFT.EQ.'H') THEN
            DO JY = YSTART,YEND,NYSTEP
              DO IX = ROW-XSTART,ROW-XEND,-1*NXSTEP
                K = ( JY - 1 )*ROW + IX
                SUM = SUM + REALBF(K)
                      AREA = AREA + 1.0
                      IXTEMP = ROW - IX
                CALL REPLACE_PIX( IXTEMP*2, JY*2 )
              END DO
            END DO
          ELSE
            DO JY = YSTART,YEND,NYSTEP
              DO IX = XSTART,XEND,NXSTEP
                K = ( JY - 1 )*ROW + IX
                SUM = SUM + REALBF(K)
                      AREA = AREA + 1.0
                CALL REPLACE_PIX( IX*2, JY*2 )
              END DO
            END DO
          END IF
          AVERAGE = SUM/AREA
          PHASE = (32767./2. - AVERAGE)/32767.*-360.
          CALL GRSBFD
          TYPE *,' BOX # = ',IBOX
          TYPE *,' AREA = ',AREA
          TYPE *,' AVERAGE PIXEL VALUE = ',AVERAGE
          TYPE *,' AVERAGE PHASE = ',PHASE

C
C and in the output file
C
          WRITE(7,702) IBOX, AREA, AVERAGE, PHASE
 702      FORMAT(I7,'| ',G11.5,'| ',G14.7,6X,'| ',G14.7)
C
C and write data to echo intensity storage array
C
          ECHO_INTENS(IBOX,IFILES) = AVERAGE
C
C increment the box number
C
          IBOX = IBOX + 1
          NDATA = BOXES( IBOX, 1 )
        END DO
        EVEN_ODD = .NOT.EVEN_ODD
C
C now set up the next in the series to be processed
C
        IF(FILE(IPERIOD+3).EQ.'9') THEN
          FILE(IPERIOD+2) = FILE(IPERIOD+2) + 1
          FILE(IPERIOD+3) = '0'
        ELSE
          FILE(IPERIOD+3) = FILE(IPERIOD+3) + 1
        ENDIF
C
CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
CCCCCCCCCC
C
        IMODE = 2
        POINT = .FALSE.
        END DO
C
        CALL GRINEND
C
        STOP
 1001   CONTINUE
        TYPE *,'****ERROR; Bad file or filename. Try again...'
        STOP
C
 600    FORMAT(A)
C
    END
```